

On the Detection of Shared Data Manipulation in Distributed Optimization

Mohannad Alkhrajah, Rachel Harris, Samuel Litchfield, David Huggins, and Daniel K. Molzahn

Abstract—This paper investigates the vulnerability of the Alternating Direction Method of Multipliers (ADMM) algorithm to shared data manipulation, with a focus on solving optimal power flow (OPF) problems. Deliberate data manipulation may cause the ADMM algorithm to converge to suboptimal solutions. We derive a sufficient condition for detecting data manipulation based on the theoretical convergence trajectory of the ADMM algorithm. We evaluate the performance of the detection condition on three data manipulation strategies with various complexity and stealth. The simplest attack sends the target values and each iteration, the second attack uses a feedback loop to find the next target values, and the last attack uses a bilevel optimization to find the target values. We then extend the three data manipulation strategies to avoid detection by the detection conditions and a neural network (NN) detection model. We also propose an adversarial NN training framework to detect shared data manipulation. We illustrate the performance of our data manipulation strategy and detection framework on OPF problems. The results show that the proposed detection condition successfully detect most of the data manipulation attacks. However, the bilevel optimization attack strategy that incorporates the detection methods may avoid being detected. Countering this, our proposed adversarial training framework detects all the instances of the bilevel optimization attack.

Index Terms—Cybersecurity, Data manipulation, Distributed optimization, Optimal power flow.

I. INTRODUCTION

Distributed optimization algorithms allow multiple agents to collaboratively solve large-scale optimization problems. Agents using distributed optimization solve subproblems iteratively and exchange the solutions of the shared variables with their neighbors at each iteration. If the solutions are correct and accurate, the algorithm converges to the optimal solution under mild technical assumptions. Using distributed algorithms to solve power system optimization problems, such as optimal power flow (OPF), has the potential to scale computations, increase reliability, and improve data privacy [1].

Since distributed algorithms rely on communicating shared variables, these algorithms are vulnerable to communication nonidealities and data manipulation. Most of the existing

literature on distributed optimization assumes that the participating agents are trustworthy and the shared data are accurate. Nonetheless, communications are prone to errors, and agents may deliberately share inaccurate solutions, e.g., to increase their profit by changing their local generators' outputs.

A. Related Work

Communication nonidealities significantly impact the performance of distributed algorithms. Even with a low probability of occurrence, large errors can prevent the algorithm from converging [2]. Malicious agents may inject random errors to execute “denial-of-service” attacks that cause the algorithm to diverge. However, making the algorithm converge to a suboptimal solution while being stealthy is more challenging and requires deliberate manipulations.

Deliberate data manipulations of distributed optimization are plausible, but limited research investigates this type of attacks. Reference [3] considers false data injection attacks on a distributed algorithm that solves OPF problems with the DC power flow approximation (DCOPF). Reference [3] also proposes a detection method that estimates the shared variables using data communicated in previous iterations. The agents then update a reputation index for the neighboring agents based on the deviation from the expected values of the shared data. An extension in [4] considers OPF problems for radial networks using a second-order cone programming relaxation. Other work in [5] proposes the use of power line communication and encryption to prevent “man-in-the-middle” attacks that compromise the communication between the agents. However, the method in [5] only detects attacks on the communication between agents but not on the agent's controllers. Moreover, the methods in [3]–[5] consider a simplistic attack scenario in which the attacker repeatedly shares data corresponding to a malicious target solution.

Our prior work introduced and analyzed various data manipulation strategies that drive distributed optimization algorithms to suboptimal solutions [6]–[8]. We consider a stealthy attack model in [8] that uses a feedback loop to reduce the deviation of the manipulated data from the expected value. We also propose an attack model that uses bilevel optimization with the other subproblems in the lower level. The bilevel optimization attack model finds a solution that optimizes the attacker's malicious objective and ensures the convergence of the algorithm in two iterations. Due to the fast convergence of this attack, reputation-based detection methods may fail since these methods require multiple iterations before detecting an attack. This paper extends our previous work by considering a new detection method and more sophisticated attack strategies.

Support from NSF AI Institute for Advances in Optimization, #2112533. Mohannad Alkhrajah is with the Computational Science Center, National Renewable Energy Laboratory, Golden, CO 80401, USA (email: mohannad.alkhrajah@nrel.gov). Rachel Harris, and Daniel K. Molzahn are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: mohannad@gatech.edu; rharris94@gatech.edu; molzahn@gatech.edu). Samuel Litchfield and David Huggins are with the Cybersecurity, Information Protection, and Hardware Evaluation Research Laboratory, Georgia Tech Research Institute, Atlanta, GA 30332 USA (e-mail: samuel.litchfield@gtri.gatech.edu; david.huggins@gtri.gatech.edu)

Using neural network (NN) models is a promising approach for detecting data manipulation [9]. Our prior work in [7], [8] shows that NN detection models have the potential to detect a variety of data manipulation strategies in the context of distributed optimization algorithms. However, using NN models for anomaly detection is challenging due to the nature of anomalies, as they are rare and heterogeneous, and thus difficult to collect on a large scale for training [9].

The detection models discussed above are data-driven with no detectability guarantees. Conversely, [10] proposes an analytical detection method for convex problems. The method in [10] estimates the Hessian matrix of the subproblems' augmented Lagrangian using the shared data. If the estimated Hessians are not positive semidefinite, which is necessary for convexity, then there is data manipulation. This method requires shared data from a large number of iterations. Moreover, numerical issues may arise due to poor matrix conditioning when estimating Hessian matrices, especially for solutions that are close to the optimal solution. Thus, the method requires shared data from a large number of iterations to improve conditioning at the expense of more computations. Nevertheless, these challenges may cause inaccurate estimates, potentially resulting in false positives. Moreover, the detection method in [10] is limited to convex problems.

B. Contributions

This paper investigates the vulnerability of the Alternating Direction Method of Multipliers (ADMM) algorithm to shared data manipulation when solving OPF problems. This paper proposes and evaluates several data manipulation and detection strategies. The main contributions of the paper are:

- (1) Development of an analytical detection condition based on the convergence trajectory of the ADMM algorithm to check the correctness of the shared data. This condition verifies that the shared data corresponds to a solution for an optimization problem consistent with the solutions from previous iterations. Unlike existing data-driven approaches, we analytically derive this detection condition. The detection condition uses the shared variables of a single agent from any three consecutive iterations. Thus, the other agents can use the condition during any intermediate iteration to identify the malicious agents. Moreover, the detection condition applies to convex and non-convex problems with non-differentiable objectives. To the best of our knowledge, this is the first detection condition with these properties.
- (2) Proposal of sophisticated data manipulation attacks that use bilevel optimization to bypass detection methods. Unlike the existing simple threat models in the literature, the proposed attack incorporates an NN detection model and the analytical detection condition that we propose in this paper. These attacks embed the detection methods into the attacker's bilevel optimization problem using mixed-integer linear programming (MILP) constraints.
- (3) Proposal of an adversarial NN training framework to improve the detectability of data manipulation. The adversarially trained NN detects the proposed data manipulation attacks and can be extended to detect new attacks by retraining the NN model to enhance detection accuracy. We show that even when

the attacker has access to the detection methods, bypassing the detection methods is computationally expensive, and thus renders the attacks ineffective. We also show that the analytical detection condition increases the efficiency of NN training by reducing the number of feasible attacks.

C. Organization

This paper is organized as follows. Section II presents the OPF problem and the ADMM algorithm. Section III proposes an analytical detection condition. Section IV describes three data manipulation attacks that bypass detection methods. Section V presents an NN training framework to detect data manipulation. Section VI presents numerical results of the proposed methods. Section VII gives conclusions.

II. BACKGROUND

This section presents the background information and notation for the OPF problem and the ADMM algorithm.

A. Optimal Power Flow

OPF is a fundamental optimization problem in power systems that finds the controller setpoints that minimize operational cost subject to the power flow equations and engineering constraints. There is a wide variety of OPF formulations, including various approximations and relaxations [11]. We present a general OPF formulation with the DC approximation for illustrative purposes, but the results in this paper apply to other OPF formulations. The DCOPF problem is

$$\min \sum_{g \in \mathcal{G}} c_{g2} (p_g^G)^2 + c_{g1} p_g^G + c_{g0} \quad (1a)$$

subject to:

$$\sum_{g \in \mathcal{G}_i} p_g^G - \sum_{l \in \mathcal{L}_i} p_l^L = \sum_{(i,j) \in \mathcal{E}} p_{ij}^E, \quad \forall i \in \mathcal{N}, \quad (1b)$$

$$p_{ij}^E = b_{ij}(\theta_i - \theta_j), \quad \forall (i,j) \in \mathcal{E}, \quad (1c)$$

$$p_g^{min} \leq p_g^G \leq p_g^{max}, \quad \forall g \in \mathcal{G}, \quad (1d)$$

$$|p_{ij}^E| \leq p_{ij}^{max}, \quad \forall (i,j) \in \mathcal{E}, \quad (1e)$$

$$\theta_r = 0, \quad (1f)$$

where \mathcal{N} , \mathcal{E} , \mathcal{G} , and \mathcal{L} are the sets of buses, branches, generators, and loads, respectively. The subsets $\mathcal{G}_i \subset \mathcal{G}$ and $\mathcal{L}_i \subset \mathcal{L}$ are the generators and loads connected to bus $i \in \mathcal{N}$. The decision variables are the buses' voltage phase angles $\theta \in \mathbb{R}^{|\mathcal{N}|}$, the generators' active power outputs $p^G \in \mathbb{R}^{|\mathcal{G}|}$, and the branches' active power flows $S \in \mathbb{R}^{|\mathcal{E}|}$, where $|\cdot|$ denotes the cardinality of a set. We denote the load demands with $p^L \in \mathbb{R}^{|\mathcal{L}|}$. We use $b_{ij} \in \mathbb{R}$ to denote the susceptance of branch $(i,j) \in \mathcal{E}$. Generator $g \in \mathcal{G}$ has a quadratic cost function with coefficients c_{g2} , c_{g1} , and c_{g0} .

The objective (1a) minimizes the generation cost. The equality (1b) enforces power balance, and (1c) defines the branches' power flow. Inequality (1d) bounds the generators' power output between p_g^{min} and p_g^{max} , and (1e) bounds the branches' power flow below p_{ij}^{max} . Equality (1f) sets the reference angle in the chosen reference bus r .

For notational simplicity, we group the variables in a vector $x = [\theta^T (p^G)^T (p^E)^T]^T$, where $(\cdot)^T$ is the transpose operator.

We denote the inequality and equality constraints in (1b)–(1f) as $h^E(x) = 0$ and $h^I(x) \leq 0$. We further define the set of feasible solutions as $\Omega = \{x | h^E(x) = 0, h^I(x) \leq 0\}$.

B. Alternating Direction Method of Multipliers

ADMM is a well-known distributed optimization algorithm based on the augmented Lagrangian method [12]. We first present the general form of the ADMM algorithm and then describe a special case that solves the consensus problem.

1) *General Form:* The ADMM algorithm solves problems in the form:

$$\min_{x,z} \quad f(x) + g(z) \quad (2a)$$

$$\text{subject to:} \quad Ax + Bz = c, \quad (2b)$$

$$x \in \mathcal{X}, z \in \mathcal{Z}, \quad (2c)$$

where $x \in \mathbb{R}^n$ and $z \in \mathbb{R}^m$ are decision variables constrained to the nonempty, closed, convex sets \mathcal{X} and \mathcal{Z} , and $A \in \mathbb{R}^{p \times n}$, $B \in \mathbb{R}^{p \times m}$, and $c \in \mathbb{R}^p$ are the consistency constraint parameters. The functions $f: \mathcal{X} \rightarrow \mathbb{R}$ and $g: \mathcal{Z} \rightarrow \mathbb{R}$ are proper convex functions. The formulation (2) depict problems with two sets of decision variables x and z that have separable objective functions and linear coupling constraints.

The ADMM algorithm uses an augmented Lagrangian function to relax the consistency constraints (2b):

$$L_\rho(x, z, y) = f(x) + g(z) + y^\top (Ax + Bz - c) + \frac{\rho}{2} \|Ax + Bz - c\|_2^2,$$

where $y \in \mathbb{R}^p$ are dual variables, $\rho \in \mathbb{R}_{>0}$ is tuning parameter, and $\|\cdot\|_2$ denotes the l_2 -norm. The ADMM algorithm solves the augmented Lagrangian problem by alternatively solving for x and z , and then updating the dual variables using a gradient ascending step. Thus, iterate $k+1$ solutions are:

$$x^{k+1} := \operatorname{argmin}_{x \in \mathcal{X}} f(x) + (y^k)^\top Ax + \frac{\rho}{2} \|Ax + Bz^k - c\|_2^2, \quad (3a)$$

$$z^{k+1} := \operatorname{argmin}_{z \in \mathcal{Z}} g(z) + (y^k)^\top Bz + \frac{\rho}{2} \|Ax^{k+1} + Bz - c\|_2^2, \quad (3b)$$

$$y^{k+1} := y^k + \rho(Ax^{k+1} + Bz^{k+1} - c). \quad (3c)$$

Defining the primal residual $r^k = Ax^k + Bz^k - c$ and the dual residual $s^k = \rho A^\top B(z^k - z^{k-1})$ at iteration k , the algorithm terminates when the norms (often l_2 or l_∞) of the primal and dual residuals are below a predefined tolerance.

2) *Consensus Problem:* Consensus problems are a special case of (2) that have multiple subproblems with separable objective functions and global variables. Let \mathcal{A} be the set of agents solving the subproblems. We denote the decision variables of agent $i \in \mathcal{A}$ as $x_i \in \mathbb{R}^n$, constrained to a convex set \mathcal{X}_i . We introduce auxiliary variables $z \in \mathbb{R}^n$ and enforce consistency between the shared variable by equating the same local variables with an auxiliary variable. The consensus optimization problem is:

$$\min_{x,z} \quad \sum_{i \in \mathcal{A}} f_i(x_i) \quad (4a)$$

$$\text{subject to:} \quad x_i - z = 0, \quad \forall i \in \mathcal{A}, \quad (4b)$$

$$x_i \in \mathcal{X}_i, \quad \forall i \in \mathcal{A}. \quad (4c)$$

Problem (4) is a special case of (2) with $f(x) = \sum_{i \in \mathcal{A}} f_i(x_i)$, $g(z) = 0$, $c = 0$, $A = I$, where I is the identity matrix of appropriate size, and B consists of $|\mathcal{A}|$ vertically stacked identity matrices multiplied by -1 . The ADMM algorithm then solves the consensus problem via solving the following:

$$x_i^{k+1} := \operatorname{argmin}_{x_i \in \mathcal{X}_i} f_i(x_i) + (y_i^k)^\top x_i + \frac{\rho}{2} \|x_i - z^k\|_2^2, \quad \forall i \in \mathcal{A}, \quad (5a)$$

$$z^{k+1} := \frac{1}{|\mathcal{A}|} \sum_{i \in \mathcal{A}} x_i^{k+1}, \quad (5b)$$

$$y_i^{k+1} := y_i^k + \rho(x_i^{k+1} - z^{k+1}), \quad \forall i \in \mathcal{A}, \quad (5c)$$

where $y_i \in \mathbb{R}^n$ are agent i dual variables.

To solve OPF problems (1) using the ADMM algorithm (5), consider the case where there are multiple agents defined by the set \mathcal{A} , each of which operates a partitioned subset of the buses $\mathcal{N}_i \subset \mathcal{N}$, $i \in \mathcal{A}$ along with the corresponding connected elements, i.e., generators \mathcal{G}_j and loads \mathcal{L}_j , $\forall j \in \mathcal{N}_i$. Although the objective function of the OPF problem (1a) is separable, the variables and constraints are coupled. To eliminate the coupling, we introduce auxiliary variables corresponding to the power flows of each branch that connects two partitions and the voltages of the boundary buses. Each agent takes a copy of the coupled constraints and variables in addition to consistency constraints between the coupled variables and the corresponding auxiliary variables. The consensus OPF problem thus has the form of (4) with $\mathcal{X}_i = \Omega_i$, $\forall i \in \mathcal{A}$, i.e., the set of OPF constraints in (1), and can be solved using the ADMM algorithm (5). The ADMM convergence guarantee is limited to convex problems, such as the DCOPF problem in (1). Nonetheless, empirical results show that the ADMM algorithm frequently converges to good solutions for non-convex OPF problems [13].

III. DETECTION CONDITION FOR ANOMALOUS SHARED DATA

This section presents a sufficient condition to detect inconsistencies in the solutions shared by the agents when using the ADMM algorithm. The detection condition uses the fact that the agents repeatedly solve the same subproblem with different parameters. We exploit the subproblem structure to derive a necessary condition for the solutions of the subproblems that only depends on the shared variables, which are not private. When an agent violates this condition, this agent must not be solving the same subproblem, thus yielding a sufficient condition to detect data manipulation. Before stating the detection condition, we have the following two assumptions:

Assumption 1. The objective function f is lower semi-continuous, but not necessarily convex, over the constraint set \mathcal{X} [14, Def. 1.5].

Assumption 2. The constraint set \mathcal{X} satisfies the linear independence constraint qualification (LICQ) at x^k for any $k \geq 1$ [15, Def. 12.4]

Using these two assumptions, we state the sufficient detection condition in the following proposition.

Proposition (Sufficient Detection Condition). Let x^k and z^k be the iterate k solutions of the ADMM algorithm (3) and the

problem satisfies assumptions 1 and 2. Define $\hat{z}^k = 2Bz^k - Bz^{k-1} - c$. If $(x^{k+1} + A^{-1}\hat{z}^k)^\top A^\top A(x^{k+1} - x^k) \geq \epsilon$ for any $k \geq 1$ and small $\epsilon \in \mathbb{R}_{>0}$, then there is data manipulation.

We derive the detection condition using the local subproblems' first-order optimality conditions. Since the agents share optimal solutions at each iteration, we can validate solution optimality by cross evaluating their objective functions with the current and previous shared variables. The complete proof of the sufficient detection condition is in the appendix.

The value of ϵ depends on the optimality tolerance of the numerical solutions. To preclude false positives, i.e., avoid flagging an attack while there is no data manipulation, we must select ϵ to be higher than the solver's numerical tolerance.

For the consensus problem (4), we evaluate the consistency parameters A , B , and c as described in Section II-B2. The condition thus becomes

$$(x_i^{k+1} - 2z^k + z^{k-1})^\top (x_i^{k+1} - x_i^k) \leq 0, \quad \forall i \in \mathcal{A}. \quad (6)$$

The two assumptions we used in the proposition are typical for NLP problems and generally true for OPF problems [16]. An implicit assumption we used is the ability of agents to solve the subproblem to global optimality. This assumption is hard to validate for non-convex problems. Nonetheless, empirical studies such as [17] show that local NLP solvers often find good solutions with small optimality gaps for many OPF instances. If an upper bound on the optimality gap is known, we can incorporate this bound in the value of ϵ when solving NLP problems and the proposition holds.

Thus, this condition is sufficient for detecting data manipulation in non-convex problems with non-differentiable objectives, only requires the values of shared variables from three consecutive iterations, and can be checked at each iteration. Moreover, since we can evaluate this condition for each agent separately, the condition identifies the agents that manipulate the shared data. To the best of our knowledge, this is the first sufficient condition with these advantages.

IV. DATA MANIPULATION WITH EMBEDDED DETECTION MODELS

The detection condition in Section III detects various types of data manipulations, as Section VI shows later in the paper. However, a sophisticated attacker with knowledge of the detection methods could incorporate the detection condition into their attack model to compute attacks that avoid detection. To analyze these sophisticated attacks, this section first reviews three data manipulation models from [8] that steer the ADMM algorithm to malicious operating points. Then, this section describes how an attacker could use the detection condition and an NN detection model to remain undetected.

A. Data Manipulation Strategies

We build on the attack model proposed in [8], which presents three data manipulation strategies that drive the results of the distributed algorithm to a malicious operating point. Throughout this section, we denote the attacker as agent a .

1) *Simple Attacker Model*: In this model, the attacker simply shares the target values of the shared variables at each iteration. This model requires finding a target point that is feasible for neighboring agents before the attack starts. The simple attack is easily detected because the attacker sends the same values in each iteration.

2) *Feedback Attacker Model*: The feedback attack model is a generalization of the simple attack model that is harder to detect. Instead of repeatedly sending the exact target values, the attacker uses a proportional–integral–derivative (PID) feedback loop to compute the shared variables. Similar to the simple attack, the attacker determines the desired target values of the shared variables in advance. Let \hat{x}_a be the attacker's target values, x_a be the shared variables that the attacker actually sends, and z_a be the shared variables defined in (5). The attacker computes the shared variables at iteration $k + 1$:

$$x_a^{k+1} = \hat{x}_a + K_p e_k + K_i \sum_{n=1}^k e_n + K_d (e_k - e_{k-1}), \quad (7)$$

where K_p , K_i , and K_d are the PID feedback tuning parameters and $e_k = \hat{x}_a - z_a^k$ is the deviation of the neighboring agents' shared variables from the attacker's target values. Choosing $K_p = K_i = K_d = 0$ yields the simple attack model.

3) *Bilevel Attacker Model*: This model involves solving a bilevel optimization that includes the subproblems of the neighboring agents in the lower level and the attacker's objective and constraints in the upper level. As Fig. 1 illustrates, the attacker computes the shared data at a selected start iteration s and the next two iterations by solving (8):

$$\min_{x,y,z} F(x_a^{s+2}) \quad (8a)$$

subject to:

$$x_i^k = \operatorname{argmin}_{x_i \in \Omega_i} f_i(x_i) + (y_i^{k-1})^\top x_i + \frac{\rho}{2} \|x_i - z_i^{k-1}\|, \quad \forall i \in \mathcal{A}_a, k \in \{s, s+1, s+2\}, \quad (8b)$$

$$z^k = \frac{1}{|\mathcal{A}|} \sum_{i \in \mathcal{A}} x_i^k, \quad \forall i \in \mathcal{A}_a, k \in \{s, s+1\}, \quad (8c)$$

$$y_i^k = y_i^{k-1} + \rho(x_i^k - z_i^k), \quad \forall i \in \mathcal{A}_a, k \in \{s, s+1\}, \quad (8d)$$

$$x_a^{s+2} \in \Omega_a, \quad (8e)$$

$$x_a^{s+2} = x_i^{s+2}, \quad \forall i \in \mathcal{A}_a, \quad (8f)$$

where $F: \Omega_a \rightarrow \mathbb{R}$ is the attacker's objective function describing how they would like to manipulate the solution of the distributed optimization algorithm (e.g., increasing local generation in the attacker's area). The set Ω_i denotes the OPF constraints for agent i as defined in (1). The set \mathcal{A}_a consists of the attacker's neighboring agents.

The lower-level problems in (8b) are the local subproblems of the neighboring agents for three consecutive iterations. The first iteration s can be calculated in advance since the values of z^{s-1} and y^{s-1} are known, but we keep this iteration in the formulation to illustrate the need to find the corresponding x_i^s , $\forall i \in \mathcal{A}_a$. Constraints (8c)–(8d) are the ADMM updates for iterations s and $s+1$. Constraints (8e) and (8f) ensure that the attacker's solution is feasible and the consensus is achieved at iteration $s+2$.

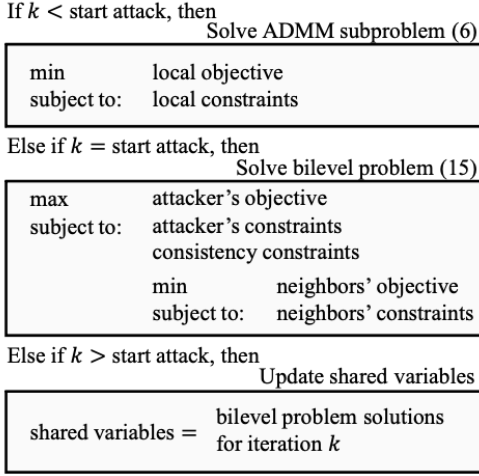


Fig. 1. The attacker's logic when using the bilevel model. The attacker selects a starting iteration. Before starting the attack, the attacker solves normal ADMM subproblems (5). After starting the attack, the attacker shares manipulated data obtained by solving the bilevel problem (8).

The complexity of the bilevel problem (8) depends on the power flow model used in the lower-level subproblems (8b) and the form of the upper-level objective (8a). With a linear power flow approximation and a linear objective function F , both the lower- and upper-level problems are linear programs. We can then solve the bilevel problem via reformulating the lower-level using the Karush–Kuhn–Tucker (KKT) conditions parameterized with the upper-level variables y and z . Then, we can use a standard big-M method [18] or a type-1 special ordered set (SOS1) method [19] to reformulate the KKT conditions as MILP constraints.

B. Embedding Detection Models

This section extends the three data manipulation strategies we previously described to incorporate the SC detection condition in Section III and an NN detection model. We do not consider the CC condition described in Section III because of its dependence on the final solution and, as we will show in Section VI, we found that the SC condition is more accurate.

1) *Embedding Detection Conditions*: To bypass the detection condition described in Section III, the attacker imposes additional constraints on their shared variables to ensure satisfying (6). For the simple and feedback attack models, a sophisticated attacker projects the shared variables computed by the attacks onto the feasible region of the detection condition (6) by solving the following problem:

$$\min_{x_a} \|x_a - \tilde{x}_a\|_2^2 \quad (9a)$$

subject to:

$$(x_a - 2z^k + z^{k-1})^\top (x_a - x_a^k) \leq 0, \quad (9b)$$

$$\tilde{x}_a = \hat{x}_a + K_p e_k + K_i \sum_{n=1}^k e_n + K_d (e_k - e_{k-1}), \quad (9c)$$

where x_a is the vector of the attacker's shared variable, z_a is the vector of shared variables as defined in (5), \hat{x}_a is

the vector of target shared variables, \tilde{x}_a is the output of the feedback attack in (7), and $e_n = \hat{x}_a^n - z_a^n$, $n \in \{1, 2, \dots, k\}$, are error terms. The objective function (9a) finds the closest shared variables to the output of the feedback attack \tilde{x}_a . Constraint (9b) ensures that the detection condition will not detect the attack, and (9c) is the output of the feedback attack.

Letting $K_p = K_i = K_d = 0$, we obtain the simple attack model with embedded detection conditions. The simple attack in this case is a projection of the attacker's target values onto the closest value that cannot be detected by the detection condition.

Due to (9b), problem (9) is a non-convex quadratic problem that the attacker can solve using an NLP solver to find a local solution. Any feasible solution to (9) that satisfies (6) will bypass the detection condition. Thus, locally optimal solutions are sufficient to drive the ADMM algorithm to the attacker's target values while avoiding detection. However, the solutions of (9) are not necessarily the best possible attack.

For the bilevel attack model, the attacker adds constraints to (8) that ensure the satisfaction of the detection condition (6). The additional constraints are

$$(x_a^k - 2z^{k-1} + z^{k-2})^\top (x_a^k - x_a^{k-1}) \leq 0 \quad (10)$$

for iterations $k \in \{s, s+1, s+2\}$, where x_a^s , x_a^{s+1} , and x_a^{s+2} are the attacker's shared variables; z_a^s and z_a^{s+1} are the shared variables as defined in (5); and x_a^{s-1} , z_a^{s-1} , and z_a^{s-2} are the shared variables from the prior iterations. As a non-convex quadratic inequality, enforcing (10) in the upper level increases the attacker's computational challenges when solving the bilevel problem (8). Nonetheless, commercial solvers like Gurobi are applicable to this type of quadratically constrained problem when appropriately formulated.

2) *Embedding Neural Network Detection Model*: Our previous work in [8] presents an NN detection model that shows promising results in detecting attacks. To avoid detection by the NN model in [8], this section presents a data manipulation attack that embeds the NN in the bilevel problem by adding constraints ensuring that the NN indicates no attack has occurred. For the simple and feedback attacks, embedding NN detection models into the attack strategies is more challenging, as we will discuss at the end of this section.

Consider an NN with a set \mathcal{U} of layers indexed from 1 to $|\mathcal{U}|$, each with a set \mathcal{V}_u , $u \in \mathcal{U}$, of neurons indexed from 1 to $|\mathcal{V}_u|$. The inputs of the NN are $O_0 \in \mathbb{R}^{|\mathcal{V}_0|}$, where \mathcal{V}_0 is the set of the inputs indexed from 1 to $|\mathcal{V}_0|$. The output consists of a single neuron as $O_{|\mathcal{U}|} \in \mathbb{R}$. The NN layers' outputs are:

$$O_u = G_u(W_u O_{u-1} + b_u), \quad \forall u \in \mathcal{U}, \quad (11)$$

where $G_u: \mathbb{R}^{|\mathcal{V}_u|} \rightarrow \mathbb{R}^{|\mathcal{V}_u|}$, $u \in \mathcal{U}$, are the non-linear activation functions, and $W_u \in \mathbb{R}^{|\mathcal{V}_u| \times |\mathcal{V}_{u-1}|}$ and $b_u \in \mathbb{R}^{|\mathcal{V}_u|}$ are the weight and bias parameters of the NN layers.

In [8], we trained an NN to detect shared data manipulation using the l_2 -norm of the primal residuals $\|r^k\|_2$ from the last 50 iterations as inputs. When the NN outputs $O_{|\mathcal{U}|} < 0$, the NN flags an attack. We chose $G_u(x) = \text{ReLU}(x) = \max\{0, x\}$ (see Fig. 2) as the activation functions for the hidden layers and the identity for the output layer, i.e., $G_{|\mathcal{U}|}(x) = x$.

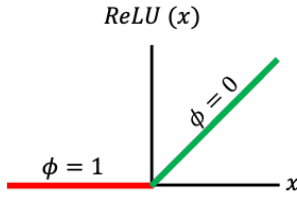


Fig. 2. $ReLU(x) = \max\{0, x\}$ with binary variable $\phi \in \{0, 1\}$, where $\phi = 1$ indicates the inactive red region and $\phi = 0$ the active green region.

Building on the work in [8], we realized that using inputs to the NN based on the SC condition (6) evaluated for multiple iterations yields a better detection accuracy compared to the primal residuals. Specifically, the NN's input at iteration s is:

$$[O_0]_v = (x_a^k - 2z^{k-1} + z^{k-2})^\top (x_a^k - x_a^{k-1}), \quad (12)$$

$$k = s - |\mathcal{V}_0| + v, \forall v \in \mathcal{V}_0.$$

To formulate this NN embedding, we use the method proposed in [20] that models an NN with activation functions $ReLU(x) = \max\{0, x\}$ as MILP constraints. We introduce binary variables $\phi_u \in \{0, 1\}^{|\mathcal{V}_u|}, \forall u \in \mathcal{U}$, to indicate the operation region of the $ReLU$ function as shown in Fig. 2. We also introduce nonnegative real variables $I_u^+ \in \mathbb{R}_{\geq 0}^{|\mathcal{V}_u|}$ and $I_u^- \in \mathbb{R}_{\geq 0}^{|\mathcal{V}_u|}, \forall u \in \mathcal{U}$, to model the inputs of the hidden layer u , with constraints that permit only one of a particular neuron's variables to take a nonzero value. For each layer, the inputs of the activation functions are $I_u^+ - I_u^- = W_u O_{u-1} + b_u$ and the outputs are $O_u = ReLU(I_u^+ - I_u^-) = I_u^+$.

Using this notation, we formulate the NN as MILP at iteration k as follows. For each hidden layer $u \in \mathcal{U} \setminus \{\mathcal{U}\}$, we enforce the following constraints:

$$I_u^+ - I_u^- = W_u O_{u-1} + b_u, \quad (13a)$$

$$O_u = I_u^+, \quad (13b)$$

$$[\phi_u]_v = 1 \rightarrow [I_u^+]_v \leq 0, \quad \forall v \in \mathcal{V}_u, \quad (13c)$$

$$[\phi_u]_v = 0 \rightarrow [I_u^-]_v \leq 0, \quad \forall v \in \mathcal{V}_u, \quad (13d)$$

$$I_u^+ \geq 0, I_u^- \geq 0. \quad (13e)$$

Constraints (13c)–(13d) are disjunctive inequalities that ensure only one of same entry of I_u^+ and I_u^- has a nonzero value. These constraints can be reformulated as an MILP using Big-M or SOS1 methods. We define the inputs of the NN using (12) and the outputs as:

$$O_{|\mathcal{U}|} = W_{|\mathcal{U}|} O_{|\mathcal{U}|-1} + b_{|\mathcal{U}|}, \quad (14a)$$

$$O_{|\mathcal{U}|} \geq \epsilon. \quad (14b)$$

Constraint (14b) ensures that the NN does not detect the outputs of the bilevel problem. A small positive number ϵ in (14b) accounts for the feasibility tolerance of the solver.

Solving the bilevel problem with an embedded NN is hard and does not scale well with large systems. Moreover, the complexity of the problem increases as the number of the hidden layers increases. Methods for efficiently embedding NNs in optimization problems has been an increasingly studied topic in recent years. Other NN embedding models may produce stronger MILP formulation than (13) [21]. In future

work, we plan to explore other NN embedding models with the bilevel optimization to scale to larger systems.

Embedding an NN to avoid detection with the simple and feedback attacks is more challenging than the bilevel optimization attack. The simple and feedback attacks compute the shared variables for a single iteration. Solving the projection problem (9) with an embedded NN ensures the next iteration is undetected. However, the inputs of the NN detection model will be dominated by manipulated shared data after a few iterations. This makes finding the next iteration's value by solving (9) with the NN embedding (12)–(14) challenging because the attacker needs to consider many iterations after the next iteration to ensure finding feasible solutions. On the other hand, the bilevel attack avoids this problem because this attack ensures termination after two iterations. We will show in Section VI that embedding an NN with the simple and feedback attacks leads to infeasible solutions.

V. ADVERSARIALLY TRAINED NEURAL NETWORK

When augmented with (10)–(14), the outputs of the bilevel optimization (8) bypass both the detection condition (6) and the NN detection model presented in Section IV-B2. To better detect such sophisticated data manipulation attacks, we develop an adversarial training framework. The framework uses the bilevel attack model to generate adversarial training samples as shown in Fig. 3. The framework's parameters are the number of initial training samples T_{train} , adversarial training iterations M_{train} , and samples for each adversarial training iteration $N_{samples}$.

Initially, we generate a dataset consisting of T_{train} attacked and unattacked samples. We then train an NN to detect the attacked samples. The trained NN detects all the attacked samples even with a small number of layers and neurons. However, the bilevel attack with an embedded NN, as described in Section IV-B2, can bypass this detection model. Accordingly, the second stage of the framework iteratively trains the NN on the output of the bilevel attack model. At each iteration, we generate $N_{samples}$ undetected attacked samples and retrain the NN on the new samples. The goal of this framework is to train the NN until the bilevel problem with an embedded NN becomes infeasible or too computationally expensive to solve, rendering this attack strategy ineffective. We focus on the bilevel attack model because the simple and feedback attacks fail to find adversarial samples.

VI. SIMULATION RESULTS

This section presents the computational results of the data manipulation strategies and detection methods when solving OPF problems using the ADMM algorithm. We first show the impacts of the attack strategies on the solutions of the ADMM algorithm. Next, we demonstrate the effectiveness of the detection methods and compare their performance. We then present the results of embedding the detection methods into the data manipulation strategies. Finally, we show the results of the proposed adversarially trained NN framework in identifying data manipulation that embed the detection methods.

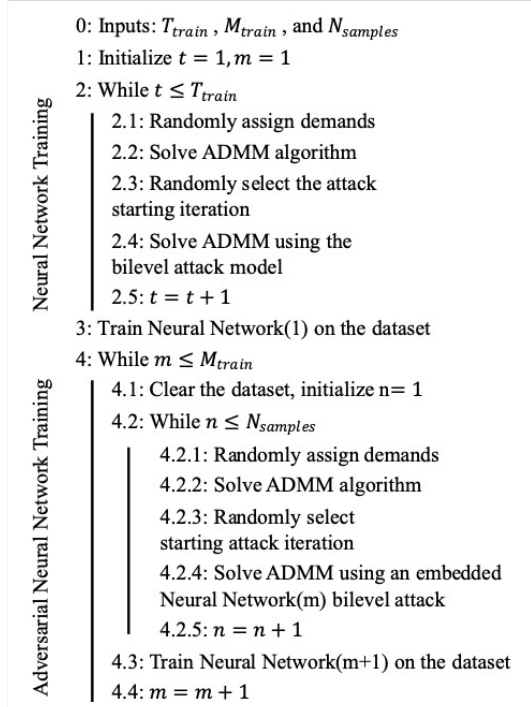


Fig. 3. Adversarial training framework flowchart consisting of two stages: (1) initial training in steps 2 and 3, and (2) adversarial training in step 4.

A. Simulation Setup

We solved OPF problems with the case3_lmbd, IEEE 14-bus, and IEEE 118-bus test cases from the PGLib-OPF library [22]. We decomposed the systems into three areas and assumed that the attacker controls one of the areas. We set the tuning parameter $\rho = 100$ and terminate the ADMM algorithm when the l_∞ -norm of the primal residuals is less than 10^{-2} .

Our implementation in Julia uses the PowerModelsADA library [23] to solve OPF problems with the ADMM algorithm and the BilevelJuMP library [24] to solve bilevel optimization problems. We trained NN models using the Flux library [25] and used Gurobi to solve all optimization problems. The simulations use an Intel Xeon CPU with 24 physical cores and 16GB memory.

B. Data Manipulation

We first present the results of the data manipulation attacks. The attacker controls one of the areas and tries to steer the output of the ADMM to a suboptimal solution that increases the attacker's local generation. The attacker uses three data manipulation strategies: (1) simple attack, (2) feedback attack, and (3) bilevel optimization attack. In both the simple and feedback attacks, the attacker finds the target value before starting the iterative process, whereas the bilevel attack finds the maximum achievable target values within the next two iterations after starting the attack.

We used 100 runs to generate the results. In each run, we randomly select the load demands between 50% and 150% of the base demands with a constant power factor. We also randomly select a starting iteration for the attack within the first hundred iterations. We use the *optimality gap*, defined as the relative change in the objective function with and without data manipulation, to measure the impacts of the attacks.

TABLE I
THE AVERAGE OPTIMAL SOLUTION OF THE TEST CASES OVER 100 RUNS AND THE OPTIMALITY GAP FROM THE THREE ATTACK STRATEGIES

Case	Optimal Solution	Simple Attack	Feedback Attack	Bilevel Attack
3-bus	4388.2	70.2%	70.2%	70.2%
14-bus	7628.3	24.4%	24.4%	29.2%
118-bus	125945.9	28.6%	28.6%	19.7%

TABLE II
ACCURACY OF THE DETECTION METHODS

case	Detection Method	No Attack ¹	Simple Attack ²	Feedback Attack ²	Bilevel Attack ²
3-bus	DC	100%	98.5%	100%	100%
	NN	100%	100%	100%	100%
14-bus	DC	100%	98.4%	100%	100%
	NN	100%	100%	100%	100%
118-bus	DC	100%	98.5%	100%	94.2%
	NN	97.8%	99.2%	99.9%	94.2%

¹True negative. ²True positive.

Table I shows the average optimal objective function value of the DCOPF problem and the average optimality gap when using each attack strategy. The three attacks find suboptimal solutions that increase the attacker's local generation. The optimality gap increases to 70% for the 3-bus system and around 20% for the 14- and 118-bus systems after the attacks.

C. Detection Methods

We present the results of detection methods and a trained NN detection model. We checked the detection conditions for all iterations until convergence. For the NN detection, our previous work [8] shows that we can achieve high detection accuracy by increasing the number of hidden layers. The work in [8] uses 8 hidden layers with 50 inputs corresponding to the mismatches of the shared variables over the last 50 iterations. Here, we show that we can maintain high detection accuracy with fewer hidden layers while considering fewer prior iterations as inputs to the NN.

As described in (12), we use a NN with 4 hidden layers and inputs from the last 10 iterations of the ADMM algorithm before convergence. We train the NN by generating 4000 samples of the three attack strategies and normal ADMM solutions. We then test the results using 1000 samples to produce the statistics shown in Table II.

Both detection conditions avoid false negatives when the value of ϵ is appropriately selected. We set $\epsilon = 0.1$ in the detection condition for the 3- and 14-bus test systems and $\epsilon = 5$ for the 118-bus system to avoid false negative, i.e., flagging an attack for unattacked samples.

The analytical detection condition and the NN detection model successfully identify most of the attacks, with accuracy above 97% for most of the test cases and often achieving above 99% accuracy. Although the NN models use 10 iterations as inputs, their performance is close to the detection condition, which uses all the shared data from the first to the last iteration as inputs. Moreover, we can enhance the accuracy of the NN models by increasing the number of hidden layers and inputs. However, there is no control over the false negative results of the NN outputs, which can be seen in the 118-bus test system.

TABLE III
SUCCESS RATE OF THE ATTACK STRATEGIES
WITH EMBEDDED DETECTION METHODS

case	Detection Method	Feedback Attack	Bilevel Attack
3-bus	DC	100%	100%
	NN	0%	72%
	NN+DC	0%	44%
14-bus	DC	100%	0%
	NN	0%	0%
	NN+DC	0%	0%
118-bus	DC	100%	0%
	NN	0%	0%
	NN+DC	0%	0%

D. Embedding Detection Methods

The high detection accuracy demonstrated in Table II can be compromised if the attacker knows the detection models. To show this, we use the two attack strategies with the embedded detection model described in Section IV. We solve the models with 100 instances of the three test systems while varying the loads. We observe that the solver may take a very long time to find a solution due to the complexity of the problem. We set a maximum time limit of 100 seconds for the bilevel attack and 5 seconds for the feedback attack (since the attacker solves the bilevel problem once, but repeatedly solves projection problems for each iteration).

The success rate of the two attacks is summarized in Table III. The results indicate that knowing the detection models is not enough to ensure a successful attack. For the 14- and 118-bus systems, the attack strategies fail to find solutions in all instances when embedding the trained NN model. Moreover, the feedback attack always bypasses the two detection conditions, while the bilevel attack fails with the 14- and 118-bus systems. On the other hand, the feedback attack fails to bypass the NN detection model for the reason we previously discussed at the end of Section IV.

E. Adversarial Training

To improve the NN detection accuracy, we use the adversarial training framework described in Section V. We use the 3-bus system to demonstrate the effectiveness of the framework because the attack models fail to avoid detection with the other two test systems. We iteratively trained the NN on the outputs of the bilevel attack models with an embedded NN. At each iteration, we solve the bilevel problem with $N_{samples} = 100$ samples and collect the successful samples to retrain the NN. We use the bilevel attack model considering only the NN model and then using both the NN and the detection condition.

The results of the adversarial training are shown in Fig. 4. The NN successfully detects almost all attacks (more than 99.9%) with $M_{train} = 16$ iterations. We also noticed that the detection accuracy improved more quickly when using the detection condition alongside the NN. Thus, the detection condition increases the efficiency of the training process by focusing on a subset of all the possible data manipulation scenarios to enhance the detection accuracy.

The results show that in a few instances, the bilevel attack finds solutions that are undetectable by the adversely trained NN. The framework does not guarantee that there is no possible bilevel attack solution. However, the possibility of

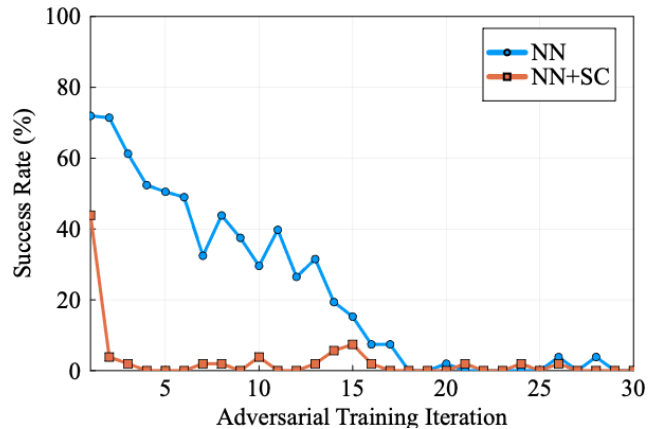


Fig. 4. Success rate of the bilevel attack with embedded detection methods over the adversarial training iterations. The blue line indicates the result of the bilevel attack with an embedded neural network (NN) and the red line with an embedded neural network and the detection condition (NN+DC).

finding a successful bilevel attack within 100 seconds is very low (i.e., less than 0.1% with the setup in this paper).

VII. CONCLUSION

Distributed algorithms have many advantages that make them appealing approaches for coordinating systems with multiple agents. Future power systems with thousands or millions of controllers may coordinate their operations by solving OPF problems using distributed algorithms. However, as we demonstrate in this paper, distributed algorithms may be vulnerable to shared data manipulation attacks that drive the algorithm to suboptimal solutions. Since agents repeatedly solve the same subproblems, we exploit the subproblem structure to detect shared data manipulations by deriving a sufficient condition. We also show that a sophisticated attacker with knowledge of the detection methods may avoid detection by embedding the detection conditions in their attacks. We then use an adversarially trained NN framework to enhance the detectability of data manipulation strategies even when the attacker knows the detection methods.

The proposed framework and results in this paper are based on embedding the detection methods into the attacker's data manipulation strategies. The embedding of the detection methods involves solving mixed-integer programs with non-convex quadratic constraints, which are computationally challenging. Accordingly, the results indicate that embedding the detection model into the attacker's problem does not scale well to large test systems. Our future work aims to develop embedding models that are more scalable through approximating or using stronger MILP models for the detection methods. Moreover, the attacker's problem is computationally difficult even without the NN embedding due to the need to ensure convergence of the distributed algorithm to the target solutions. Thus, our ongoing work aims to develop more scalable attack models that ensure convergence of the distributed algorithm.

APPENDIX

PROOF OF THE SUFFICIENT DETECTION CONDITION

Proof. To derive the detection condition, let $\bar{f}(x) := f(x) + \delta_{\mathcal{X}}(x)$, the objective function of the local subproblem with the indicator function $\delta_{\mathcal{X}}(x)$, where $\delta_{\mathcal{X}}(x) = 0$ if $x \in \mathcal{X}$ and

$\delta_{\mathcal{X}}(x) = \infty$ otherwise. Let $\bar{L}_{\rho}(x, z, y) := \bar{f}(x) + y^{\top}(Ax + Bz - c) + \frac{\rho}{2}\|Ax + Bz - c\|_2^2$, the augmented Lagrange function (3a) with the indicator function $\delta_{\mathcal{X}}(x)$. At iteration $k + 1$, we have $x^{k+1} := \operatorname{argmin} \bar{L}_{\rho}(x, z^k, y^k)$. Since f is lower semi-continuous over the set \mathcal{X} , and \mathcal{X} satisfies the LICQ at x^{k+1} [15, Def. 12.4]. Since x^{k+1} minimizes $\bar{L}_{\rho}(x, z^k, y^k)$, then $0 \in \partial \bar{L}_{\rho}(x^{k+1}, z^k, y^k)$, the subgradient of the objective function of the local subproblem evaluated at x^{k+1} [14, Theorem 8.15]. Thus,

$$\begin{aligned} 0 &\in \partial \bar{L}_{\rho}(x^{k+1}, z^k, y^k), \\ &= \partial[\bar{f}(x^{k+1}) + (y^k)^{\top}(Ax^{k+1} + Bz^k - c) \\ &\quad + \frac{\rho}{2}\|Ax^{k+1} + Bz^k - c\|_2^2], \\ &= \partial \bar{f}(x^{k+1}) + A^{\top}(y^k) + \rho A^{\top}(Ax^{k+1} + Bz^k - c), \\ &= \partial \bar{f}(x^{k+1}) + A^{\top}(y^k) \\ &\quad + \rho A^{\top} \left[\frac{1}{\rho}(y^{k+1} - y^k) - Bz^{k+1} + c + Bz^k - c \right], \\ &= \partial \bar{f}(x^{k+1}) + A^{\top}(y^{k+1}) - \rho A^{\top}(Bz^{k+1} - Bz^k), \end{aligned}$$

where the fourth equality uses the dual update (3c) to substitute for $Ax^{k+1} = \frac{1}{\rho}(y^{k+1} - y^k) - Bz^{k+1} + c$. This implies that x^{k+1} also minimizes $\bar{f}(x) + (y^{k+1})^{\top}Ax - \rho(Bz^{k+1} - Bz^k)^{\top}Ax$. Thus, we have

$$\begin{aligned} \bar{f}(x^{k+1}) + (y^{k+1})^{\top}Ax^{k+1} - \rho(Bz^{k+1} - Bz^k)^{\top}Ax^{k+1} &\leq \\ \bar{f}(\tilde{x}) + (y^{k+1})^{\top}A\tilde{x} - \rho(Bz^{k+1} - Bz^k)^{\top}A\tilde{x}, \end{aligned} \quad (15)$$

for any $\tilde{x} \in \mathbb{R}^n$. Replacing k with $k - 1$, we have

$$\begin{aligned} \bar{f}(x^k) + (y^k)^{\top}Ax^k - \rho(Bz^k - Bz^{k-1})^{\top}Ax^k &\leq \\ \bar{f}(\tilde{x}) + (y^k)^{\top}A\tilde{x} - \rho(Bz^k - Bz^{k-1})^{\top}A\tilde{x}, \end{aligned} \quad (16)$$

for any $\tilde{x} \in \mathbb{R}^n$. Letting $\tilde{x} = x^k$ in (15), $\tilde{x} = x^{k+1}$ in (16), and combining both inequalities, we obtain the following:

$$\begin{aligned} \bar{f}(x^{k+1}) + (y^{k+1})^{\top}Ax^{k+1} - \rho(Bz^{k+1} - Bz^k)^{\top}Ax^{k+1} \\ + \bar{f}(x^k) + (y^k)^{\top}Ax^k - \rho(Bz^k - Bz^{k-1})^{\top}Ax^k &\leq \\ \bar{f}(x^k) + (y^{k+1})^{\top}Ax^k - \rho(Bz^{k+1} - Bz^k)^{\top}Ax^k + \\ + \bar{f}(x^{k+1}) + (y^k)^{\top}Ax^{k+1} - \rho(Bz^k - Bz^{k-1})^{\top}Ax^{k+1}. \end{aligned}$$

The values of local objective functions with the local constraints $\bar{f}(x^k)$ and $\bar{f}(x^{k+1})$, which are private information, cancel out, and only the shared variables remain. Rearranging the terms and substituting for the dual variables, we obtain

$$\begin{aligned} (y^{k+1})^{\top}(Ax^{k+1} - Ax^k) - \rho(Bz^{k+1} - Bz^k)^{\top}(Ax^{k+1} - Ax^k) \\ - (y^k)^{\top}(Ax^{k+1} - Ax^k) \\ + \rho(Bz^k - Bz^{k-1})^{\top}(Ax^{k+1} - Ax^k) &\leq 0, \\ (y^{k+1} - y^k)^{\top}(Ax^{k+1} - Ax^k) \\ - \rho(Bz^{k+1} - 2Bz^k + Bz^{k-1})^{\top}(Ax^{k+1} - Ax^k) &\leq 0, \\ \rho(Ax^{k+1} + Bz^{k+1} - c)^{\top}(Ax^{k+1} - Ax^k) \\ - \rho(Bz^{k+1} - 2Bz^k + Bz^{k-1})^{\top}(Ax^{k+1} - Ax^k) &\leq 0, \\ (Ax^{k+1} + 2Bz^k - Bz^{k-1} - c)^{\top}(Ax^{k+1} - Ax^k) &\leq 0. \end{aligned} \quad (17)$$

We use the dual update (3c) in the third inequality to substitute $y^{k+1} - y^k = \rho(Ax^{k+1} + Bz^{k+1} - c)$. Factoring A and

substituting for $\hat{z}^k = 2Bz^k - Bz^{k-1} - c$ yields the detection condition. \square

REFERENCES

- [1] D. K. Molzahn, F. Dörfler, H. Sandberg, S. H. Low, S. Chakrabarti, R. Baldick, and J. Lavaei, "A survey of distributed optimization and control algorithms for electric power systems," *IEEE Trans. Smart Grid*, vol. 8, no. 6, pp. 2941–2962, 2017.
- [2] M. Alkhrajah, C. Menendez, and D. K. Molzahn, "Assessing the impacts of nonideal communications on distributed optimal power flow algorithms," *Electric Power Syst. Res.*, vol. 212, p. 108297, 2022, presented at 22nd Power Syst. Comput. Conf. (PSCC 2022).
- [3] J. Duan, W. Zeng, and M. Chow, "Resilient distributed DC optimal power flow against data integrity attack," *IEEE Trans. Smart Grid*, vol. 9, no. 4, pp. 3543–3552, 2018.
- [4] Z. Cheng and M. Chow, "Resilient collaborative distributed AC optimal power flow against false data injection attacks: A theoretical framework," *IEEE Trans. Smart Grid*, vol. 13, no. 1, pp. 795–806, 2022.
- [5] Y. Yang, G. Raman, J. Peng, and Z. Ye, "Resilient consensus-based AC optimal power flow against data integrity attacks using PLC," *IEEE Trans. Smart Grid*, vol. 13, no. 5, pp. 3786–3797, 2022.
- [6] R. Harris, M. Alkhrajah, D. Huggins, and D. K. Molzahn, "On the impacts of different consistency constraint formulations for distributed optimal power flow," in *Texas Power and Energy Conf. (TPEC)*, 2022.
- [7] R. Harris and D. K. Molzahn, "Detecting and mitigating data integrity attacks on distributed algorithms for optimal power flow using machine learning," in *57th Hawaii Int. Conf. Syst. Sci. (HICSS)*, 2024.
- [8] M. Alkhrajah, R. Harris, S. Litchfield, D. Huggins, and D. K. Molzahn, "Analyzing malicious data injection attacks on distributed optimal power flow algorithms," in *54th North American Power Symp. (NAPS)*, 2022.
- [9] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection: A review," *ACM Comput. Surv.*, vol. 54, no. 2, 2021.
- [10] E. Munsing and S. Moura, "Cybersecurity in distributed and fully-decentralized optimization: Distortions, noise injection, and ADMM," arXiv:1805.11194, 2018.
- [11] D. K. Molzahn and I. A. Hiskens, "A survey of relaxations and approximations of the power flow equations," *Found. Trends Electric Energy Syst.*, vol. 4, no. 1-2, pp. 1–221, 2019.
- [12] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, 2010.
- [13] A. Kargarian, Y. Fu, and Z. Li, "Distributed security-constrained unit commitment for large-scale power systems," *IEEE Trans. Power Syst.*, vol. 30, no. 4, pp. 1925–1936, 2015.
- [14] R. T. Rockafellar and R. J. Wets, *Variational Analysis*. Springer Science & Business Media, 2009, vol. 317.
- [15] J. Nocedal and S. J. Wright, *Numerical Optimization*. Springer, 1999.
- [16] A. Hauswirth, S. Bolognani, G. Hug, and F. Dörfler, "Generic existence of unique lagrange multipliers in AC optimal power flow," *IEEE Control Syst. Lett.*, vol. 2, no. 4, pp. 791–796, 2018.
- [17] S. Gopinath and H. L. Hijazi, "Benchmarking large-scale ACOPF solutions and optimality bounds," in *IEEE Power & Energy Society General Meeting (PESGM)*, 2022.
- [18] J. Fortuny-Amat and B. McCarl, "A representation and economic interpretation of a two-level programming problem," *J. Oper. Res. Soc.*, vol. 32, no. 9, pp. 783–792, 1981.
- [19] J. P. Vielma and G. L. Nemhauser, "Modeling disjunctive constraints with a logarithmic number of binary variables and constraints," *Math. Prog.*, vol. 128, pp. 49–72, 2011.
- [20] M. Cacciola, A. Frangioni, and A. Lodi, "Structured pruning of neural networks for constraints learning," arXiv:2307.07457, 2023.
- [21] R. Anderson, J. Huchette, W. Ma, C. Tjandraatmadja, and J. P. Vielma, "Strong mixed-integer programming formulations for trained neural networks," *Math. Prog.*, vol. 183, no. 1-2, pp. 3–39, 2020.
- [22] IEEE PES Task Force on Benchmarks for Validation of Emerging Power System Algorithms, "The Power Grid Library for benchmarking AC optimal power flow algorithms," arXiv:1908.02788, Aug. 2019.
- [23] M. Alkhrajah, R. Harris, C. Coffrin, and D. K. Molzahn, "PowerModelsADA: A framework for solving optimal power flow using distributed algorithms," *IEEE Trans. Power Syst.*, vol. 39, no. 1, pp. 2357–2360, 2024.
- [24] J. D. Garcia, G. Bodin, and A. Street, "BilevelJuMP.jl: Modeling and solving bilevel optimization in Julia," arXiv:2205.02307, 2022.
- [25] M. Innes *et al.*, "Fashionable modelling with Flux," arXiv:1811.01457, 2018.