On the Impacts of Different Consistency Constraint Formulations for Distributed Optimal Power Flow

Rachel Harris*, Mohannad Alkhraijah*, David Huggins†, and Daniel K. Molzahn*
*School of Electrical and Computer Engineering, Georgia Institute of Technology
†Georgia Tech Research Institute, Georgia Institue of Technology

***Abstract*—The optimal power flow (OPF) problem finds the least costly operating point which meets the power grid's operational limits and obeys physical power flow laws. Complementing today's centralized optimization paradigm, future power grids may rely on distributed optimization where multiple agents work together to determine an acceptable operating point. In distributed algorithms, local agents solve subproblems to optimize their region of the system and share data to achieve consistency with their neighboring agents' subproblems. This paper investigates how different methods of enforcing power flow consistency constraints between local areas in distributed optimal power flow impact convergence rate and a classifier's ability to detect malicious cyberattack. The distributed OPF problem is solved with the alternating direction method of multipliers (ADMM) algorithm. First, the ADMM algorithm's convergence rate is compared for three different consistency constraint formulations. Next, the paper considers a cyberattack in which the integrity of information shared between agents is compromised, causing the algorithm to exhibit unacceptable behavior. A support vector machine (SVM) classifier is trained to detect the presence of manipulated data from such cyberattacks. Results demonstrate that consistency constraint formulation impacts the classifier's detection performance; for certain formulations, detection is highly accurate.**

***Index Terms*—Distributed optimization, smart grid, optimal power flow, cyber threat, cyber security**

I. Introduction

The electric power grid is undergoing fundamental changes in power generation, storage, and transmission. The widespread integration of distributed energy resources (DERs), increasing network complexity and the need for privacy motivate the use of distributed control and optimization methods to complement traditional centralized operation [1]. Distributed algorithms assign portions of the problem to multiple computing agents, reducing the computational burden of each agent. These agents solve their subproblems in parallel so that large-scale problems can be solved more quickly. In addition, parameters in the cost function and system constraints remain private to individual agents.

The optimal power flow (OPF) problem finds the least costly operating point for a power system which obeys physical power flow laws and meets grid operational constraints. Solving OPF problems in a distributed manner requires decomposing the power system into local areas and formulating constraints to enforce physical power flow consistency across all areas. This paper investigates the impact of constraint formulation on algorithm convergence rate and malicious attack detectability. Consistency constraints should support fast convergence so that large-scale OPF problems can be solved within minutes on the future smart grid. In addition, cyberattacks are a significant threat to the emerging smart grid [2]. A distributed algorithm may be vulnerable to attacks in which a local agent or communication link is compromised and an attacker manipulates the shared data.

This paper compares three different methods proposed in previous work for enforcing consistency constraints between local areas in the power network. Section II summarizes existing literature and outlines the paper's contributions. Section III formulates the distributed optimization algorithm, the consistency constraint methods, and the cyberattack model by which the adversary steers the solution to some desired target. Section IV formulates the support vector classification model for attack detection. Section V compares convergence rate and malicious attack detection performance for the three consistency constraint methods. The results are based on distributed OPF algorithms implemented with the three consistency constraint formulations. The number of algorithm iterations required to reach convergence is compared for several test cases. Next, the classifier's attack detection precision, recall and accuracy are computed for each method to assess and compare performance. Section V-C provides details on how these statistics are computed. The results show that engineers should consider both convergence rate and attack detectability when deciding how to enforce consistency between local areas for power system distributed optimization.

II. Background and Literature Review

Distributed algorithms such as the popular alternating direction method of multipliers (ADMM) [3], [4] and the auxiliary problem principle (APP) [6] have been successfully applied to optimal power flow. Any distributed algorithm must enforce power flow consistency between local areas. This paper considers three different methods for enforcing consistency proposed in previous work. The first method duplicates the voltage phasors across the tie lines [3]–[5]. The second method adds dummy buses at the endpoints of the tie lines [6]–[8], while the third method splits areas through the buses [9]–[11]. Section III-A provides more details on these methods. This paper characterizes how these methods differ in convergence rate and impact a classifier's ability to detect malicious attack. To the best of our knowledge, there has not yet been a comparison between these possible methods in previous literature.

Past work on distributed OPF cyberattacks demonstrated the financial threat posed by an adversary who steers the solution to some desired target [12]. Our previous work formulated multiple attack models and analyzed how quickly and effectively they converged to the target operating point [15]. Another study devised an algorithm resilient to such attacks in which agents use information propagated from their two-hop neighbors to detect and replace suspicious data [13]. This is a promising method, but is only demonstrated for one type of attack on a static network. To provide adequate protection against cyber threats, defensive strategies should be robust to changes in the network and perform well against multiple types of attacks.

This paper uses a machine learning classifier to flag suspicious data. Machine learning for smart grid cybersecurity is not a new concept; such methods have been used, for instance, to detect false data injection attacks on system measurements [14]. However, there has been little exploration of machine learning for detection of economic attacks in distributed optimal power flow. This paper makes the following contributions:

1) Characterization of the impact of different consistency constraint formulations on convergence rates.
2) Use of a machine learning based detection algorithm to identify attacks.
3) Analysis of the impact of different consistency constraint formulations with respect to cybersecurity.

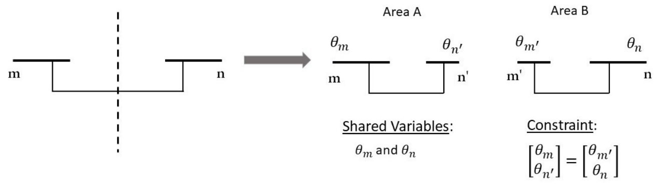
For the sake of simplicity and notational brevity, this paper applies distributed optimization to the DC optimal power flow problem. Each of the three methods for enforcing consistency could be generalized to the AC power flow equations as well as various power flow relaxations and approximations by extending the variables that are shared between agents. Thus, in addition to showing valuable results for the widely used DC power flow approximation, this paper provides a starting point for our future work in generalizing the results presented here to other power flow formulations.

III. Proposed Method

This section formulates the consistency constraints and ADMM distributed optimization algorithm used to solve the DC OPF, as well as the cyberattack model.

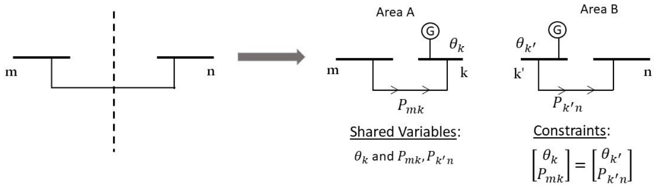
A. Problem Formulation

The OPF problem seeks to optimize performance while meeting operational limits and obeying physical power flow laws. In this paper, the objective is to minimize generation cost, and the power flow law is the linearized DC approximation.



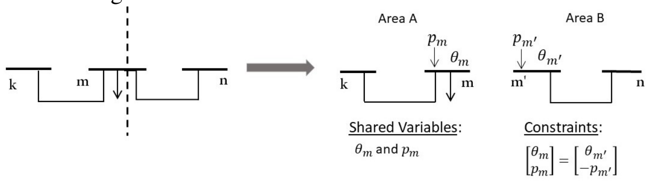
Area A
Area B
θ_m
$\theta_{n'}$
$\theta_{m'}$
θ_n
m
n
m
n'
m'
n
Shared Variables:
θ_m and θ_n
Constraint:
$\begin{bmatrix}\theta_m\\ \theta_{n'}\end{bmatrix} = \begin{bmatrix}\theta_{m'}\\ \theta_n\end{bmatrix}$

(a) Method 1: Split across tie-line; add copy of neighboring bus



Area A
Area B
G
θ_k
$\theta_{k'}$
m
n
m
k
k'
n
P_{mk}
$P_{k'n}$
Shared Variables:
θ_k and $P_{mk}, P_{k'n}$
Constraints:
$\begin{bmatrix}\theta_k\\ P_{mk}\end{bmatrix} = \begin{bmatrix}\theta_{k'}\\ P_{k'n}\end{bmatrix}$

(b) Method 2: Split across tie-line; insert dummy bus with artificial generation



(c) Method 3: Split through the bus; each area keeps a copy of the bus

Fig. 1: Three Decomposition Methods

1) Consistency Constraints: To decompose the OPF problem across multiple computing agents, the power system is divided into m regions. Consistency constraints ensure valid power flow at the boundaries. As shown in Figure 1, three different constraint formulations are considered in this paper. The first method splits the system through the tie-line and gives boundary buses copies of their neighbor's phase angle to ensure they agree. In method 2, a dummy bus is inserted at the break in the tie-line with artificial generation to allow for power injection from a neighboring area. Neighboring agents must agree on the phase angle at and power flow through the dummy bus. For method 3, the system is split through a bus rather than the tie-line. The phase angle at all copies of the bus must be identical and the net power injection must be zero.

2) Distributed Optimization: Agents seek to minimize generation cost in their region while obeying OPF constraints. Let the areas under control of separate agents be numbered $m = 1, 2, .., n$. Let G_m denote the set of generators in area m, each with cost function $f_g(p_g)$, real power generation p_g, and generation lower limit P_g^{min} and upper limit P_g^{max}. Let S be an index set for all shared variables with $S_m \subset S$ containing indices of shared variables with copies kept by agent m. For any of agent m's shared variables x_s, $s \in S_m$, there is a corresponding $\bar{x}_s$ computed by minimizing the difference between copies of x_s across all agents k for which $s \in S_k$. For method 1, the shared variables are the phase angles at boundary buses. For method 2, the shared variables consist of phase angles at and power flow through the dummy bus inserted at the tie-line break. For method 3, the shared variables are the phase angles and net power injections at buses through which the system was split.

In this paper, the objective is to minimize generation cost. The consistency constraints are relaxed with the augmented Lagrangian technique, and the iterative ADMM algorithm is used to find the optimal solution. The subproblem solved by agent m at iteration k is given in equations (1a)-(1e).

$$\begin{aligned}
\min_{p_g^{k+1},\theta^{k+1},x_s^{k+1}} \quad & \sum_{g\in G_m} f_g(p_g^{k+1}) + \sum_{s\in S_m} \lambda_s^k(\bar{x}_s^k - x_s^{k+1}) \\
& + \frac{\rho}{2}(\bar{x}_s^k - x_s^{k+1})^2 \quad \text{(1a)}\\
\text{s.t.} \quad & p_{g,i}^{k+1} - p_{d,i} = \sum_{j\in N_m} P_{ij}^{k+1}, \qquad \forall i \in N_m \quad \text{(1b)}\\
& P_{ij}^{k+1} = B_{ij}(\theta_i^{k+1} - \theta_j^{k+1}), \ \forall (i,j)\in L_m \quad \text{(1c)}\\
& p_g^{min} \le p_g^{k+1} \le p_g^{max}, \qquad \forall g \in G_m \quad \text{(1d)}\\
& -P_{ij}^{max} \le P_{ij}^{k+1} \le P_{ij}^{max}, \ \forall (i,j)\in L_m \quad \text{(1e)}
\end{aligned}$$

N_m is the set of buses in area m, B_{ij} is the susceptance of line (i,j), L_m is the set of lines in area m and $P_{ij,max}$ is the maximum active power flow along branch (i,j). The first term in the objective function (1a) minimizes the generation cost for the agent and the latter terms correspond to the augmented Lagrangian formulation. Constraint (1b) enforces power balance at each bus, constraint (1c) defines the line power flow, constraint (1d) enforces generation limits, and constraint (1e) enforces line flow limits.

After each agent solves their subproblem, agents send their computed shared variables and the corresponding dual variables to their neighbors. Then, each agent m updates the $\bar{x}_s$ variables according to (2).

$$\min_{\bar{x}_s^{k+1}} \quad \sum_{s\in S} \lambda_s^k(\bar{x}_s^{k+1} - x_s^{k+1}) + \frac{\rho}{2}(\bar{x}_s^{k+1} - x_s^{k+1})^2 \tag{2}$$

Finally, each agent m updates their Lagrange multipliers as shown in (3).

$$\lambda_s^{k+1} = \lambda_s^k + \rho(\bar{x}_s^{k+1} - x_s^{k+1}). \tag{3}$$

Thus the iterative algorithm alternates between minimizing the local agents' subproblems in (1), minimizing the difference between shared variable $\bar{x}_s$ and all local copies of x_s in (2), and updating dual variables in (3). The stopping criterion is the mismatch between local shared variable values x_s and the values $\bar{x}_s$ computed with neighboring agents' data. At iteration k, each agent m records the norm of the mismatches across all its shared variables as shown in (4).

$$\epsilon = \sqrt{\sum_{s\in S_m} (\bar{x}_s^k - x_s^k)^2} \tag{4}$$

The algorithm terminates when each agent reports mismatches ϵ less than some tolerance ϵ_0.

B. Attack Model

This paper considers attacks by an adversary who seeks to steer the solution to a specific target, perhaps for economic gain. We determine the attacker's target operating point by solving a system-wide OPF problem with a certain generator's active power output constrained to the desired value. The attacker saves the results which fall within its local area as the target variable values and uses one of two methods proposed in [15] to guide the final result to the desired target.

The attacker's simplest strategy is to directly send the desired values for shared variables to neighboring agents. However, this is simple to detect. For a more subtle attack, the attacker can employ a proportional-integral-derivative (PID) control method. For each shared variable x^k at iteration k, the attacker computes the error e^k as the difference between the target value and the current value from its neighbors:

$$e^k = x^k_{target} - x^k.$$

Then, the attacker computes a correction term using three tuned parameters: the proportional gain k_p, the derivative gain k_d, and the integral gain k_i. After adding this correction term, the computed shared variable value for the next iteration is

$$x^{k+1} = x^k + k_p e^k + k_d(e^k - e^{k-1}) + k_i \sum_{i=1}^{k} e^i.$$

IV. Attack Detection

We use a trained support vector machine (SVM) to classify a sequence of shared variable mismatches as either normal or under attack. Support vector classification (SVC) performs well in high-dimensional spaces [20], which is desirable for attack detection because a significant number of mismatches must be recorded before the difference in attacked mismatch trajectories becomes clear. SVM implementations are also memory efficient compared to algorithms such as k-nearest neighbors, which is important for algorithms which could be deployed in embedded computing systems on the smart grid. In addition, SVC provides a user-controlled regularization parameter that can help prevent over-fitting by reducing decision surface complexity [20]. The binary SVC algorithm seeks a hyperplane which divides the two classes of vectors with a maximum distance between the plane and the nearest vector of either class [21]. Let the training vectors be written as $\mathbf{x}_i \in \mathbb{R}^n$ for $i = 1, ..., m$, each corresponding to a label $y_i \in \{1, -1\}$. For the attack detection in this paper, each training vector $\mathbf{x}_i$ consists of the shared variable mismatches from 100 consecutive iterations of the distributed OPF algorithm. The label for each training vector is $y_i = 1$ if the shared variable mismatches are from an algorithm under attack, and $y_i = -1$ if the mismatches are from an algorithm which is not under attack. The separating hyperplane is defined by solving (5).

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{m}\xi_i \\ \text{s.t.} \quad & y_i(\mathbf{w}\cdot\phi(\mathbf{x}_i)+b) \geq 1-\xi_i \\ & \xi_i \geq 0,\ i=1,...,m \end{aligned} \tag{5}$$

Here, $\mathbf{w}$ is the vector normal to the hyperplane which separates positive and negative training samples. The slack variables ξ_i allow for non-separable data sets in which no boundary can perfectly divide the positive and negative training samples. Thus nonzero ξ_i variables represent training errors, which are penalized by adding the term $C\sum_{i=1}^{m}\xi_i$ to the objective function. The regularization parameter $C > 0$ controls the magnitude of the penalty on such errors. Larger values of C thus produce a more complex decision surface which results in greater accuracy during training but also potential over-fitting. The function $\phi(\mathbf{x})$ maps data from $\mathbb{R}^n$ to some higher dimensional space H, allowing for nonlinear SVMs in which the decision function is not a linear function of the training data. In practice, the mapping $\phi(\mathbf{x})$ is not explicitly defined; rather, a kernel function $K(\mathbf{x}_i,\mathbf{x}_j)$ such that $K(\mathbf{x}_i,\mathbf{x}_j) = \phi(\mathbf{x}_i)\cdot\phi(\mathbf{x}_j)$ is used to compute dot products in the higher dimensional space H. For this paper, we use the popular Gaussian radial basis kernel $K(\mathbf{x}_i,\mathbf{x}_j) = \exp(-\gamma||\mathbf{x}_i-\mathbf{x}_j||^2)$, where the parameter $\gamma > 0$ defines the radius of influence of individual training samples on the resulting decision function.

Standard SVM techniques solve the dual of the problem in (5) to obtain the decision function

$$f(\mathbf{x}) = \text{sgn}(\sum_{i=1}^{m} y_i\alpha_i K(\mathbf{x}_i,\mathbf{x})+b)$$

where $\text{sgn}(x)$ denotes the signum function and α_i represents the dual variables.

V. Numerical Results

This section describes the implementation and the power system test cases used to evaluate algorithm performance. Next, the numerical results for convergence rate and attack detection are presented.

A. Implementation and Case Description

The distributed optimization algorithm is implemented in Julia using the PowerModels library [16] to parse power system test case data and the JuMP optimization library [17] with the Gurobi solver. Inputs to the algorithm include the power network data in Matpower format [18] and a predefined division of the system into separate areas.

The convergence rate is compared for the three different consistency constraint formulations on the IEEE 14-bus system [19] divided into 3 and 4 areas and the IEEE 118-bus system [19] divided into 4 and 5 areas. For attack detection, the paper focuses on the IEEE 14-bus system with 4 areas. The malicious attacker takes control of agent 1 and seeks to increase output at generator 2 from 38 MW to 100 MW, increasing the total generation cost from \$7642.59 to \$8767.82. For the PID attack, the tuned parameters described in section III-B are $k_p = 0.4$, $k_d = 0.005$ and $k_i = 0.001$. These parameters were selected after exploring values in the range $10^{-4} - 1$ to find those that enabled the algorithm to converge to the desired target with shared variable mismatches most similar to those under no attack.

To generate training data for attack detection, 150 different test cases were created from the original power network. New cases were created by perturbing the active power demand at each load to be k times its original value, where k is a randomly selected value between 0 and 2. The distributed OPF algorithm was run three times on each of the 150 variable-load test cases: first under normal operation, second under the simple attack, and third under the PID attack. For each run, the time at which the attack begins was randomly chosen to be some iteration between 1 and 700.

B. Results

1) Convergence Rate: Using the DC OPF results computed using PowerModels [16], we validated the solutions obtained using the distributed optimization algorithm for all test cases. Convergence rates for the three consistency constraint formulations are compared by recording the number of algorithm iterations required to achieve average shared variable mismatch less than 10^{-4} radians for phase angles, and 10^{-4} per unit (with base power of 100 MVA) for power flows and power injections. Figure 2 shows the number of iterations performed before convergence for each method on the four test cases, and shows the total computation time in seconds at the top of each bar.

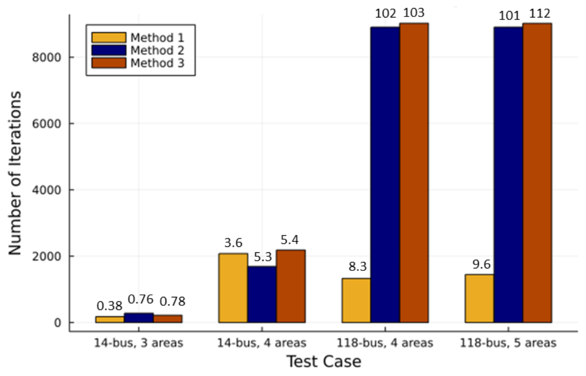
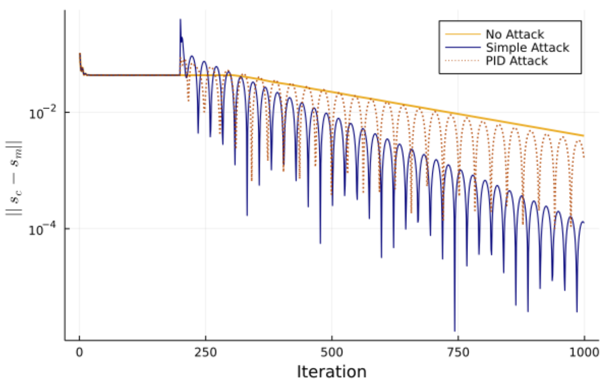
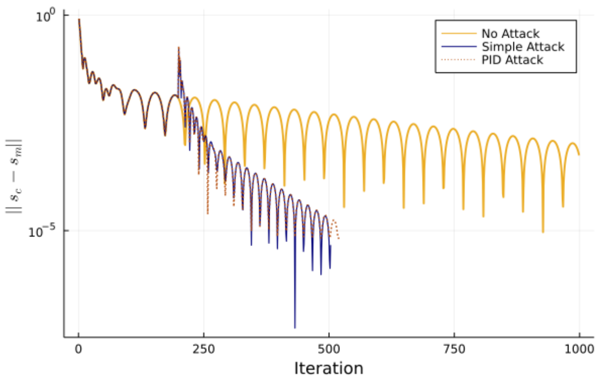


Fig. 2: Number of iterations required for convergence for three network decomposition methods, with total computation time in seconds above each bar.

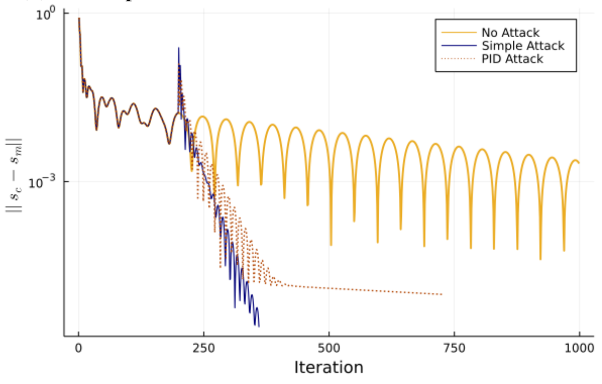
Based on these empirical results, as the size of the system increases, method 1 converges more quickly than methods 2 and 3. This is likely because method 1 builds in the most overlap between regions as copies of all neighboring buses are included in the local agent's system. In contrast, method 2 adds entirely new variables to the system without this overlap when it inserts a dummy bus at the tie-line break. Method 3 splits areas at the bus, requiring more complicated power injection computations as part of the consistency constraints. Therefore, method 1 provides the most overlap between regions with the simplest constraint formulation.



(a) Decomposition method 1 shared variable mismatches



(b) Decomposition method 2 shared variable mismatches



(c) Decomposition method 3 shared variable mismatches

Fig. 3: Shared variable mismatches for three network decomposition methods

2) Attack Detection Performance: The trajectory of the average shared variable mismatches follow different patterns under normal operation and under attack. The mismatches from the first 1000 iterations for normal operation and the two attack models are plotted in Figure 3.

Therefore, machine learning can be used to flag unusual behavior and thus detect a malicious attack. The shared variable mismatches are recorded for 1000 iterations of the algorithm. The detection process operates on vectors of 100 shared variable mismatch values. Every ten iterations, the last 100 shared mismatch values are sent to the classifier, which determines whether or not they indicate malicious attack. Thus the classifier operates on values from iterations 1-100, then 11-110, then 21-120, and so on across the 1000 algorithm iterations. As described in section V-B, we use 150 test cases and record data from solving the distributed optimal power flow three times on each test case: under no attack, simple attack, and PID attack. Since there are 91 100-iteration mismatch vectors extracted at 10-iteration intervals from each test case, the result is 40,950 input vectors, each labeled either "1" for attacked or "-1" for not attacked. Sixty percent of these vectors are used for training and forty percent are reserved for testing the trained classifier.

The attack detection performance is recorded for each of the three different consistency constraint methods. There are three statistics of interest for the detection performance: precision, recall, and total accuracy. These are computed as

$$precision = \frac{tp}{tp+fp}, \quad recall = \frac{tp}{tp+fn}$$

$$accuracy = \frac{tp+tn}{tp+fp+fn+tn}$$

where tp denotes true positives, fp denotes false positives, fn denotes false negatives and tn denotes true negatives as defined in Table 1. Thus precision decreases when normal operation is misclassified as attacked and recall decreases when a real attack is not detected. Practically, a detection mechanism with a low precision statistic leads to wasted effort reacting to false alarms. On the other hand, a low recall statistic means that some attacks are undetected, resulting in economic loss.

TABLE I: Definition of performance statistics

	Attacked	**Normal**
Classified as Attacked	true positive	false positive
Classified as Normal	false negative	true negative

The results for testing the trained classifier are shown in Table 2. The attack detection performs best for method 2 at 98% accuracy. The precision for methods 1 and 3 is lower than ideal at 92%, which means there are some false alarms in which the classifier confuses normal algorithm behavior with a system under attack. While method 1 has the best convergence rate, attack detection on method 1 is inferior in both precision and recall compared to the other two methods. The observed difference is likely caused by the sudden rapid convergence to the attacker's target solution for methods 2 and 3 when the attack begins; the shared variable mismatches soon become orders of magnitude lower than they would be under no attack. However, the method 1 mismatch magnitudes under attack do not exhibit such marked difference from those under no attack. Our ongoing work includes defining why the difference between method 1 mismatch magnitudes under attack and under no attack is less significant. The results motivate future work to improve detection for method 1 so that algorithms can operate with fast convergence as well as highly accurate attack detection.

C. Discussion and Future Work

This paper compared convergence rate and malicious attack detectability for distributed DC optimal power flow

TABLE II: Attack Detection Performance

	Precision	Recall	Accuracy
Method 1	0.92	0.96	0.93
Method 2	0.96	0.99	0.98
Method 3	0.92	0.99	0.95

under three different network decomposition methods. Numerical results demonstrate that as the system size increases, method 1 converges in the fewest number of iterations. This paper also showed that an SVM classifier can be trained to identify OPF algorithms under attack with high accuracy. The classifier exhibits best detection performance for algorithms with network decomposition method 2. Since method 1 is the best choice for convergence rate, future work should improve the detection mechanism for method 1. The list below summarizes how the attack detection proposed in this paper could be improved and used to develop resilient distributed algorithms.

1) Improved Detection Performance for Method 1: Adding features such as dual variable trajectories to the shared variable mismatches could produce a more robust detection system. In addition, a more complex mechanism could train a different decision function for different algorithm segments, since the shared variable trajectories look very different at the beginning of the iterative algorithm than at the end.

2) System Variability: This paper did not investigate how well detection generalizes to network changes such as line or generator outages. In the future, the classifier could be trained with data from test networks adjusted to reflect typical power system contingencies.

3) Attack Mitigation: Although a method to *detect* an attack was presented, no method to *mitigate* the effects of the attack by preventing solution manipulation has been described. To develop a resilient distributed OPF algorithm, it must be determined which agent has been compromised and how to replace the false data. For the ADMM algorithm presented in this paper, this may require developing an estimator for the true shared variable values.

VI. Conclusion

Distributed algorithms for operation of the emerging smart grid must be made resilient to cyber-attack. This paper has demonstrated a distributed optimal power flow algorithm and investigated how different methods of enforcing consistency between local areas impact convergence rate and a defender's ability to detect malicious attacks. Three different consistency constraint formulations were tested. The formulation denoted as "method 1" in this paper, in which agents are given copies of their neighbors' voltage angles and must agree on their values, converged most quickly. However, the classifier did not perform as well at detecting attacks on method 1 compared to the other methods. Future work will seek to improve the detection mechanism to make it highly accurate for method 1. In addition, future work will include generalizing the detection method to system changes and developing strategies to mitigate the attack's financial threat. The resulting resilient distributed optimization algorithms can be deployed on future smart grids with low risk of economic loss from a malicious attack.

References

[1] D. K. Molzahn et al., "A Survey of Distributed Optimization and Control Algorithms for Electric Power Systems," in *IEEE Trans. Smart Grid*, vol. 8, no. 6, pp. 2941-2962, Nov. 2017.

[2] T. Nguyen, M. Alhazmi, M. Nazemi, A. Estebsari and P. Dehbhanian, "Electric Power Grid Resilience to Cyber Adversaries: State of the Art," in *IEEE Access*, vol. 8, pp. 87592-87608, 2020.

[3] Y. Wang, L. Wu and S. Wang, "A Fully-Decentralized Consensus-Based ADMM Approach for DC-OPF With Demand Response," in *IEEE Trans. Smart Grid*, vol. 8, no. 6, pp. 2637-2647, Nov. 2017.

[4] K. Sun and X. A. Sun, "A Two-Level ADMM Algorithm for AC OPF With Global Convergence Guarantees," in *IEEE Trans. Power Syst.*, vol. 36, no. 6, pp. 5271-5281, Nov. 2021.

[5] W. Lu, M. Liu, S. Lin and L. Li, "Fully Decentralized Optimal Power Flow of Multi-Area Interconnected Power Systems Based on Distributed Interior Point Method," in *IEEE Trans. Smart Grid*, vol. 33, no. 1, pp. 901-910, Jan. 2018.

[6] D. Hur, J.-K. Park and B. H. Kim, "Evaluation of Convergence Rate in the Auxiliary Problem Principle Distributed Optimal Power Flow," in *Proc. Inst. Elect. Eng. Gen. Transm. Distrib.*, vol. 149, no. 5, pp. 525-532, Sep. 2002.

[7] G. Moon, Y. Wi, K. Lee and S. Joo, "Fault Current Constrained Decentralized Optimal Power Flow Incorporating Superconducting Fault Current Limiter (SFCL)," in *IEEE Trans. Appl. Supercond.*, vol. 21, no. 3, pp. 2157-2160, June 2011.

[8] N. Huebner, Y. Rink, M. Suriyah and T. Leibfried, "Distributed AC-DC Optimal Power Flow in the European Transmission Grid with ADMM," in *2020 55th Int. Univ. Power Eng. Conf. UPEC*, 2020, pp. 1-6.

[9] C. Feng, Z. Li, M. Shahidehpour, F. Wen, W. Liu and X. Wang, "Decentralized Short-Term Voltage Control in Active Power Distribution Systems," *IEEE Trans. Smart Grid*, vol. 9, no. 5, pp. 4566-4576, Sept. 2018.

[10] T. R. Alsenani and S. Paudyal, "Distributed Approach for Solving Optimal Power Flow Problems in Three-phase Unbalanced Distribution Networks," *2018 Australas. Univ. Power Eng. Conf. (AUPEC)*, 2018, pp. 1-6.

[11] T. AlSkaif and G. van Leeuwen, "Decentralized Optimal Power Flow in Distribution Networks Using Blockchain," *2019 2nd Int. Conf. Smart Energy Syst. Technol. (SEST)*, 2019.

[12] J. Duan, W. Zeng and M. Chow, "Economic Impact of Data Integrity Attacks on Distributed DC Optimal Power Flow Algorithm," *2015 N. Am. Power Symp. (NAPS)*, 2015, pp. 1-7.

[13] J. Duan, W. Zeng and M. Chow, "Resilient Distributed DC Optimal Power Flow Against Data Integrity Attack," in *IEEE Trans. Smart Grid*, vol. 9, no. 4, pp. 3543-3552, July 2018.

[14] M. Ozay, I. Esnaola, F. T. Yarman Vural, S. R. Kulkarni and H. V. Poor, "Machine Learning Methods for Attack Detection in the Smart Grid," in *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 8, pp. 1773-1786, Aug. 2016.

[15] M. Alkhraijah, S. Litchfield, C. Raslawski, D. Huggins, and D.K. Molzahn, "Analyzing Malicious Data Injection Attacks on Distributed Optimal Power Flow Algorithms," submitted for publication, 2021.

[16] C. Coffrin, R. Bent, K. Sundar, Y. Ng and M. Lubin, "PowerModels. JL: An Open-Source Framework for Exploring Power Flow Formulations," *2018 Power Syst. Comput. Conf. (PSCC)*, 2018, pp. 1-8.

[17] I. Dunning, J. Huchette and M. Lubin, "JuMP: A Modeling Language for Mathematical Optimization," in *SIAM Review*, vol. 59, no. 2, pp. 295-320, 2017.

[18] R. Zimmerman, C. Murillo-Sánchez and R. Thomas, "MATPOWER: Steady-State Operations, Planning, and Analysis Tools for Power Systems Research and Education," in *IEEE Trans. Power Syst.*, vol. 99, pp. 1-8, 2011.

[19] Power systems test case archive. [Online]. Available: https://labs.ece.uw.edu/pstca/.

[20] C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," in *Data Mining and Knowledge Discovery*, vol. 2, pp. 1-43, 1998.

[21] K. R. Muller, S. Mika, G. Ratsch, K. Tsuda and B. Scholkopf, "An Introduction to Kernel-Based Learning Algorithms," in *IEEE Trans. Neural Netw.**, vol. 12, no. 2, pp. 181-201, March 2001.