FLiER: Practical Topology Error Correction Using Sparse PMUs
C. Ponce D. S. Bindel

Abstract—In this paper, we present FLiER, or Fingerprint Linear Estimation Routine for Line Failures. FLiER is an efficient algorithm to identify single-line topology errors in power networks using readings from sparsely-deployed phasor measurement units (PMUs). When a power line is removed from a network, the event leaves a unique "voltage fingerprint" of bus voltage changes that we can identify using only the portion of the network directly observed by the PMUs. The naive brute-force algorithm to identify a failed line from such voltage fingerprints, though simple and accurate, is slow. We derive an approximate algorithm based on a local linearization that is faster and only slightly less accurate. We also extend this technique to the identification of multiple simultaneous line failures using linear programming. We present experimental results using the IEEE 57-bus and 118-bus networks.

I. INTRODUCTION

Topology error correction is an important component of a network monitoring and control system. Standard state estimation techniques assume that the computed topology is correct [1], and incorrect network topologies can result in poor and even dangerous control actions [2], [3]. In addition, topology changes due to failed lines put stress on the remaining lines and may destabilize the network. Thus, it is important to identify failed lines quickly in order to take appropriate control actions.

Currently, it is difficult to find failed lines in distribution networks. Often, utilities lack the monitoring equipment to directly detect line failures, and are unaware of issues until customers call to report power losses. The radial nature of many distribution networks makes the task somewhat easier, but distributed generation will eliminate much of this benefit. Consequently, distribution network state estimation techniques such as [4], [5], [6] are particularly vulnerable to topology changes, as less information is available to correct for them. Algorithms for failed line identification allow utilities to quickly detect such problems, and allow for accurate state estimation even when the topology changes.

Finding line failures is usually simple in transmission networks, as transmission lines have sensors that directly report failures (or switch open/closed status). However, if a sensor malfunctions, or is compromised by a cyber-intrusion, then finding the failed line is again difficult. Although failure to correctly identify a failed line is less common in a transmission network, the stakes are higher: state estimation based on incorrect topology assumptions can lead to incorrect estimates, causing operators to overlook system instability, and in the worst case, leading to avoidable blackouts. Thus, it is important to have more than one way to monitor network topology.

Most previous work on topology estimation explicitly models the circuit breakers at buses suspected of status errors [7]. Several papers solve a constrained optimization problem and use Lagrange multiplier values to detect errors [8], [9]. Others use an MW-only model of the power network, exploiting the computational advantages of the linear approximation [3], [10], [11]. Still other authors simultaneously estimate state and topology [12], [13], [10]. Other approaches include using genetic algorithms to find the correct topology [14] and using auto-associative neural networks that “resonate” in response to the correct topology [15].

In this paper, we introduce topology estimation techniques that use data collected from phasor measurement units (PMUs). PMUs are a new type of sensor that can directly measure voltage and current phasors at a bus dozens of times per second. Given knowledge
of a bus’s voltage phasor, the current phasors of the
incoming lines, and the network’s admittance properties,
one can compute the voltage phasors of neighboring
buses as well [16]. While most PMUs today are in the
transmission grids, distribution grids are likely to see
more PMUs in the future as well [17], [18].

We present here an efficient method to identify failed
d power lines in networks with a (possibly small) number
of PMUs. We assume that a complete state estimate is
obtained shortly before a line fails, e.g. through conven-
tional SCADA measurements, and we use discrepancies
between this state estimate and PMU measurements to
identify failures. Our method does not require complete
observability from PMU data; it performs well even
when there are few PMUs in the network, though having
more PMUs does improve the accuracy.

Though this paper presents the algorithm in terms of
identifying failed lines, it equally applies to the case
of identifying a newly-connected line in the network.
In fact, one could envision a range of possible FLIER
variants, each designed to address a different type of
change in the network’s admittance matrix.

II. PROBLEM FORMULATION AND FINGERPRINTS

Let \( y_{ik} = g_{ik} + jb_{ik} \) denote the elements of the
network admittance matrix \( Y \in \mathbb{C}^{n \times n} \). Furthermore, let
\( P_\ell \) and \( Q_\ell \) denote the real and reactive power injections
at a bus \( \ell \) in the network, and let \( v_\ell = |v_\ell| \exp(j\theta_\ell) \)
denote the voltage phasor at bus \( \ell \). These quantities are
related by the power flow equations

\[
H(v; Y) - s = 0
\]

where

\[
\begin{bmatrix}
H_\ell \\
H_{n+\ell}
\end{bmatrix}
= \sum_{h=1}^{n} |v_\ell||v_h|
\begin{bmatrix}
g_{ih} & b_{ih} & \cos(\theta_{ih}) \\
-b_{ih} & g_{ih} & \sin(\theta_{ih})
\end{bmatrix},
\]

with \( \theta_{ih} = \theta_\ell - \theta_h \) and

\[
s = \begin{bmatrix}
P_1 & \cdots & P_n & Q_1 & \cdots & Q_n
\end{bmatrix}^T.
\]

We note that the function \( H \) is quadratic in the first
argument, but linear in the second.

Let \((i, k)\) denote the line from bus \(i\) to bus \(k\). When
a line \((i, k)\) fails, the admittance matrix changes to

\[
Y_{ik} = Y + \Delta Y_{ik},
\]

where

\[
\Delta Y_{ik} = \begin{bmatrix}
e_i & e_k
\end{bmatrix}
\begin{bmatrix}
y'_{ii} & y'_{ik} \\
y'_{ki} & y'_{kk}
\end{bmatrix}
\begin{bmatrix}
e_i & e_k
\end{bmatrix}^T.
\]

We assume that the failure of one line does not cause
catastrophic instability in the network, and that the state
shifts from one quasi steady-state to another. As it does
this, the voltage vector shifts from \(v\) to another vector \(\hat{v}\).
Here we make a key assumption: customer loads and
real power demands at generators do not change after
the line failure. That is, we assume the complex power
injection remains constant at each load bus, the real
power injection and voltage magnitude remain constant
at generator buses, and the complex voltage phasor
remains constant at the slack bus. We believe this to be
a reasonable assumption, as generally customers do not
change their load in response to a network line failure
(barring a power outage), and generators cannot quickly
change their power output. We can also accommodate
known nonzero changes to the specified parameters.

While we do not know all of \(\hat{v}\), we do know the
voltage phasors rendered observable by the PMUs. Let \(m\)
denote the number of observable voltage phasors, and let
\( \bar{E} \in \{0, 1\}^{m \times n} \) be a matrix that selects these phasors,
i.e. the \(i\)th row of \(\bar{E}\) has a one in the position of the
observed bus and zeros elsewhere. Let \( \Delta \bar{v} \equiv \hat{v} - v \) be
the change in the vector of voltage phasors after line failure;
then \( \bar{E} \Delta \bar{v} \) denotes the known voltage phasor changes.
This set of observable voltage phasor changes acts as a
fingerprint for the change that occurred in the network.
Even when we cannot directly observe the change, we
can often identify it from the fingerprint.

The problem, therefore, is the following: given the
network state prior to a line failure and the voltage
fingerprint \( \bar{E} \Delta \bar{v} \in \mathbb{R}^m \), determine which line failed.

It is possible that two or more line failures have either
the same or practically indistinguishable fingerprints. For
example, one of two parallel lines with equal admittance
may fail, or two lines that are distant from all PMUs but
near each other may yield similar fingerprints. Often,
even when a line failure is not observable, our method
still produces valuable information. When multiple lines
have the same effect on the network, our technique can
be used to identify a small set of potential lines to inspect
more closely.

III. DERIVATION

As a brute force approach to identifying line fail-
ures, we could compute for each line \((i, k)\) the voltage
change \(\Delta v_{ik}\) that would result if \((i, k)\) failed. We would
then search all lines to find the \((i, k)\) that minimizes
\(\|\bar{E} \Delta v_{ik} - \bar{E} \Delta \bar{v}\|\). This algorithm is exact in the absence
of measurement error, fingerprint collisions, or unknown
changes in power supply or demand. However, it requires
that we solve the nonlinear power flow equations once
for each line in the network, and this is expensive. We
now present an approximate version of this algorithm
that retains high accuracy at much lower cost.
When a line \((i, k)\) fails, the admittance matrix changes by \(\Delta Y_{ik}\). The voltage and power injections then shift according to the equation
\[
H(v + \Delta v_{ik}; Y + \Delta Y_{ik}) - (s + \Delta s) = 0.
\]
(5)

Now we find \(\delta v_{ik} \approx \Delta v_{ik}\) by linearizing in \(v\) only. The linearized equation is
\[
\tilde{J}_{ik} \delta v_{ik} = r_{ik} - \delta s = 0,
\]
(6)
where
\[
r_{ik} = s - H(v; Y + \Delta Y_{ik}) = -H(v; \Delta Y_{ik})
\]
\[
\tilde{J}_{ik} = \frac{\partial H(v; Y + \Delta Y_{ik})}{\partial v}.
\]
This is equivalent to taking one step of Newton iteration on the power flow equations, starting from the old voltage vector as an initial guess.

If bus \(\ell\) is a generator bus, then \(\delta|v_\ell| = 0\), and \(Q_\ell\) is not of interest for our method, so we remove row and column \(n + \ell\) from the system of equations. We may also remove the rows and columns associated with the slack bus, as its voltage does not change after failure. Thus, if there are \(G\) generator buses and \(L\) load buses, eliminating known quantities from (6) leads to the reduced linear system
\[
J_{ik} \delta v_{ik} = r_{ik} + \delta s
\]
where \(\delta v_{ik} \in \mathbb{R}^{2L+G}\) is the reduced version of \(\delta v_{ik}\), and similarly with \(\tilde{J}_{ik}\) and \(r_{ik}\).

The term \(\delta s\) includes all the effects of changes to specified quantities such as real and reactive power at load buses or (via columns of \(\tilde{J}_{ik}\)) changes to voltage magnitudes at generator buses. Through the remainder of the paper, we assume these quantities are unchanged since the last full state estimate, i.e. \(\delta s = 0\). In some cases, we might want to relax the assumption that the bus specifications do not change; for example, a line outage may cause some load buses in a distribution network to lose power, in which case their power injection becomes zero. Smart meters on homes could then report which homes have lost power. We leave this as future work.

Given \(\delta v_{ik}\) as in (7), we use \(E \delta v_{ik}\) as an estimate for the changes in observed bus voltages were \((i, k)\) to fail, where \(E\) here represents the reduced version of \(E\). This leads us to Algorithm 1.

Algorithm 1 solves one linear problem per line. Though the Jacobian \(J_{ik}\) is different for each line, we can modify Algorithm 1 to compute \(\delta v_{ik}\) at the cost of one linear solve per bus with the pre-failure Jacobian, along with a small amount of work per line. We now show how to accomplish this.

\[
\text{Algorithm 1 Hybrid algorithm for finding failed lines.}
\]
\begin{algorithm}
\begin{algorithmic}
    \For {each node \(i\) in the network}
    \For {each node \(k\) neighboring \(i\)}
    \If {\(t_{ik}\) already computed}
    \State Continue.
    \EndIf
    \State Compute \(E \delta v_{ik}\) via Equation (7).
    \State Set \(t_{ik} = \|E \delta v_{ik} - E \Delta v\|\).
    \EndFor
    \EndFor
\end{algorithmic}
\end{algorithm}

Let \(J = \partial H(v, Y)/\partial v\) denote the pre-failure Jacobian, and note that we may write each \(\tilde{J}_{ik}\) as
\[
\tilde{J}_{ik} = \frac{\partial H(v; Y + \Delta Y_{ik})}{\partial v} = \frac{\partial H(v, Y) + \partial H(v, \Delta Y_{ik})}{\partial v} = \tilde{J} + \tilde{A}_{ik}.
\]
(8)
The vector \(H(v; \Delta Y_{ik})\) has only four nonzero entries:
\[
\begin{align*}
\hat{P}_i & = H_i = P_i + g_i' |v_i|^2, \\
\hat{Q}_i & = H_{i+n} = \hat{Q}_{ik} = \hat{Q}_{ki} = \hat{Q}_{kk} = b_k' |v_k|^2,
\end{align*}
\]
where
\[
\begin{bmatrix}
\hat{P}_i \\
\hat{Q}_i \\
\hat{Q}_{ki} \\
\hat{Q}_{kk}
\end{bmatrix} =
\begin{bmatrix}
|v_i||v_k| & g_i' & b_k' & \cos(\theta_{ik}) \\
|v_i||v_k| & b_k' & g_i' & \sin(\theta_{ik}) \\
& & & \\
& & &
\end{bmatrix}.
\]

and \(\hat{P}_{ki}\), \(\hat{Q}_{ki}\) are defined similarly. Let
\[
C_{ik} = \frac{\partial (\hat{P}_i, \hat{P}_k, \hat{Q}_i, \hat{Q}_{ki})}{\partial (\theta_i, \theta_k, |v_i|, |v_k|)} \in \mathbb{R}^{4 \times 4},
\]
by the chain rule, we can write \(C_{ik} = W_{ik} V_{ik}^T\) where
\[
W_{ik} = \frac{\partial (\hat{P}_i, \hat{P}_k, \hat{Q}_i, \hat{Q}_{ki})}{\partial (\theta_i, \theta_k, \log |v_i|, \log |v_k|)} \in \mathbb{R}^{4 \times 3},
\]
\[
V_{ik} = \frac{\partial (\theta_i, \theta_k, \log |v_i|, \log |v_k|)}{\partial (|v_i|, |v_k|)} \in \mathbb{R}^{3 \times 4}.
\]
More concretely, we have
\[
W_{ik} =
\begin{bmatrix}
-\hat{Q}_i - b_i' |v_i|^2 & \hat{P}_i + g_i' |v_i|^2 & \hat{P}_i - g_i' |v_i|^2 \\
\hat{Q}_k & \hat{Q}_k & \hat{Q}_k \\
\hat{Q}_k & \hat{Q}_k & \hat{Q}_k \\
-\hat{P}_i - g_i' |v_i|^2 & \hat{Q}_i + b_i' |v_i|^2 & \hat{Q}_i - b_i' |v_i|^2
\end{bmatrix}
\]
\[
V_{ik}^T =
\begin{bmatrix}
1 & -1 & 0 & 0 \\
0 & 0 & |v_i|^{-1} & 0 \\
0 & 0 & 0 & |v_k|^{-1}
\end{bmatrix}.
\]
Because $H(v, \Delta Y_{ik})$ does not depend on any voltage phasors other than those at nodes $i$ and $j$, we may write

$$A_{ik} = \tilde{U}_{ik} C_{ik} \tilde{U}_{ik}^T,$$

where

$$\tilde{U}_{ik} = \begin{bmatrix} e_i & e_k & e_i & e_k \end{bmatrix} \in \mathbb{R}^{2n \times 4}. \quad (10)$$

Therefore, we have

$$\tilde{J}_{ik} = \tilde{J} + \tilde{U}_{ik} W_{ik} V_{ik}^T U_{ik}^T. \quad (11)$$

Now, as in (7), we consider the reduced Jacobian

$$J_{ik} = J + U_{ik} W_{ik} V_{ik}^T U_{ik}^T = \tilde{J} + W_{ik} V_{ik}^T,$$

where $U_{ik}$ is $\tilde{U}_{ik}$ with inactive rows removed, and

$$W_{ik} = U_{ik} W_{ik} \quad V_{ik} = U_{ik} V_{ik}. \quad (13)$$

We note that we can also write $r_{ik} = W_{ik} p$:

$$r_{ik} = W_{ik} \begin{bmatrix} 0 \\ -1/2 \\ -1/2 \end{bmatrix} = -U_{ik} \begin{bmatrix} P_i \\ P_k \\ Q_i \\ Q_k \end{bmatrix}. \quad (14)$$

By the Sherman-Morrison-Woodbury formula,

$$J_{ik}^{-1} = J^{-1} - J^{-1} W_{ik} (I + X_{ik})^{-1} V_{ik}^T J^{-1}, \quad (15)$$

where $X_{ik} = V_{ik}^T J^{-1} W_{ik}$. Therefore,

$$E \delta v_{ik} = E J_{ik}^{-1} W_{ik} p$$

$$= E J^{-1} W_{ik} (I - (I + X_{ik})^{-1} X_{ik}) p$$

$$= E J^{-1} W_{ik} (I + X_{ik})^{-1} p. \quad (16)$$

To evaluate (16), we precompute $E J_{ik}^{-1}$, which involves $m$ rows of $J^{-1}$, as well as the matrices $X_{ik}$, each of which involves sixteen elements of $J^{-1}$ in the same sparsity pattern as seen in $A_{ik}$. In evaluating $X_{ik}$, we consider $U_{ik}$, $W_{ik}$, and $V_{ik}$ separately, as one can reuse rows of $U_{ik}^T J^{-1}$ for other lines involving buses $i$ or $k$. This leads us to Algorithm 2 in which precomputing these matrices is the only expensive step.

While Algorithm 2 is an improvement over re-factoring $J_{ik}$ for each line, this approach still requires that we compute (though not store) all the elements of $J^{-1}$. We can avoid this costly step by noting that, if we know that the algorithm will never select a line incident to node $i$, then we never need to compute any elements of the rows of $J^{-1}$ associated with node $i$ (unless it is part of $E J^{-1}$). Therefore, if we can filter out lines that Algorithm 2 will definitely not select, then we do not need to compute as much of $J^{-1}$.

**Algorithm 2** Faster approximation algorithm for identifying line failure.

```
Compute and store $E J^{-1}$ as well as the required sparse elements of $J^{-1}$.
for each node $i$ in the network do
    for each node $k$ neighboring $i$ do
        if $t_{ik}$ already computed then
            Continue.
        end if
        Compute $E \delta v_{ik}$ via Equation (16).
        Set $t_{ik} = \|E \delta v_{ik} - E \delta v\|$.
    end for
end for
```

Select line $(i, k)$ with minimum $t_{ik}$.

In order to quickly filter out lines from consideration, we observe from (16) that $E \delta v_{ik}$ belongs to the three-dimensional linear space

$$U_{ik} = \text{range}(E J^{-1} W_{ik}). \quad (17)$$

Thus, the error $t_{ik} = \|E \delta v_{ik} - E \delta v\|$ that appears in Algorithm 2 is bounded from below by

$$\tau_{ik} = \min_{x \in U_{ik}} \|E \Delta v - x\| = \|E \Delta v - U_{ik}\|. \quad (18)$$

Thus, if $t_{ik} < \tau_{ik'}$ for lines $(i, k)$ and $(i', k')$, then we know that $t_{ik} < \tau_{ik'}$, and we need not compute $t_{ik'}$ at all. Computing $\tau_{ik}$ for any line $(i, k)$ is cheap; exploiting this fact leads to the FLiER method (Algorithm 3), which is significantly faster than Algorithm 2.

**A. M-FLiER: An extension to multiple line failures**

The FLiER algorithm may be extended to identify multiple line failures. We consider multiple line failures to be **sequential** if the system returns to steady-state operation in between the failures; otherwise, the failures are **simultaneous**. To find sequential line failures, one may repeatedly use Algorithm 2, but if simultaneous failures occur, identifying the lines that failed is more challenging. One could check every pair or triple of lines that failed, using a simple modification to Algorithm 5. However, this may be too expensive.

When one line fails, we have the fingerprint $E \delta v_{ik} = E J^{-1} W_{ik} \ell_{ik}$. When multiple lines fail, we have

$$E \delta v = E J^{-1} \sum_{(i,k) \in L} W_{ik} \ell_{ik}. \quad (19)$$

1Example Python code of this algorithm can be found at http://www.cs.cornell.edu/~cponce/FLiER/default.htm
Algorithm 3 FLiER. Fast approximation algorithm for identifying line failure.

- Compute and store $EJ^{-1}$.
- Compute $\tau_{ik}$ for all lines $(i,k)$ via Equation (18).
- Order the lines ascending by their $\tau_{ik}$’s.
- Compute $M := t_{j_1 k_1}$, where $(j_1, k_1)$ is first.
- Set $L := (i_1, k_1)$.
- for $\ell = 2, 3, \ldots, m$ do
  - if $M < \tau_{i_\ell k_\ell}$ then
    - Continue.
  - end if
  - Compute $t_{i_\ell k_\ell}$.
  - if $t_{i_\ell k_\ell} < M$ then
    - Set $M := t_{i_\ell k_\ell}$ and $L := (i_\ell, k_\ell)$.
  - end if
- end for
- Select line $L$.

where $L$ is the set of lines in the power network. This allows us to write the following linear program:

$$\min \left( \sum_{(i,k) \in L} \|\ell_{ik}\|_{\infty} \right) + \omega \left( \|E \Delta v - E J^{-1} \tilde{z} \|_1 \right)$$

s.t.

$$\tilde{z} = \sum_{(i,k) \in L} W_{ik} \ell_{ik}.$$  \hspace{1cm} (20)

In this linear program, we use the $\ell_\infty$ norm on each $\ell_{ik}$ so that it is free to place more weight within a given line after some weight has already been placed there. We then take an “$\ell_1$ norm over lines” to encourage sparsity of line selection. The second term encourages the linear program to try to match the observed fingerprint.

In our experiments, $\omega$ value on the order of $10^{-4}$ tend to work well on the IEEE 57-bus test network. By thresholding, one may then select lines that the linear program suggests are of interest.

IV. EXPERIMENTS

Our standard experimental setup is as follows. For each line $(i,k)$ in the network, we perform a test in which we compute the system state before and after the failure of $(i,k)$, i.e. with and without $(i,k)$ included in the network model. We then pass to FLiER both the full pre-failure state and the subset of the post-failure state that would be observable by some set of PMUs.

We run most of our experiments with three PMU arrangements. In the first arrangement, we have a single PMU in the network, a near-worst-case scenario. In the second, we have a sparse PMU deployment, demonstrating what one might expect in a realistic scenario. Finally, we simulate PMUs everywhere, in which case any error is purely due to the linear approximation.

Note that the failure of lines (32, 33) and (35, 36) both cause the power flow solver to fail to converge, and so these tests are not included in the experiments.

A. Filter Effectiveness

The cost of FLiER depends strongly on the effectiveness of the filtering procedure. In Figures 1-2, we show how often the filter saves us from performing the $t_{ik}$ computation in experiments on the IEEE 57-bus and 118-bus networks. Each figure shows three PMU deployments, as described above. For each PMU deployment, we show the cumulative distribution function of the fraction of lines for which $t_{ik}$ need not be computed for each line failure.

In our experiments, the filter is effective even for modest PMU deployments. Note that both the dotted green line, the “practical scenario,” and the dashed blue line, the “complete coverage scenario,” illustrate deployments in which the number of PMUs is proportional to the network size. We show a typical case in which the filter performs well in Figure 3.

B. Accuracy

In Figures 4, 5, and 7, we show the accuracy of FLiER on the IEEE 57-bus test network. In these plots, each
Fig. 2: Cumulative distribution function for IEEE 118-bus network.

Fig. 3: A typical example of filtering effectiveness in Algorithm 3. Each column represents a line checked. Blue dots are the lower bounds $t_{ik}$, while red squares are true scores $t_{ik}$. Columns are sorted by $t_{ik}$. In this case, $t_{ik}$ only needs to be computed for eight lines.

Fig. 4: Test of our algorithm on the IEEE 57-bus network with PMUs everywhere. Each column represents one test. The black dots are $t_{ik}$’s for incorrect lines. The green dot, yellow triangle, or red square is $t_{ik}$ for the correct line.

Fig. 5: Test of our algorithm on the IEEE 57-bus network with PMUs on buses 4, 13, and 34.

column represents the $t_{ik}$ values that were computed for one line failure scenario. The black dots represent the $t_{ik}$’s of lines that get past the filter, while the green circle, yellow triangle, or red square represents $t_{ik}$ for the correct answer. If there is a green circle, then our algorithm correctly identified the actual line that failed. If there is a yellow triangle, the correct line was not chosen but was among the top three lines selected by the algorithm. Otherwise, there is a red square.

In Figure 4 we show results from a test carried out with PMUs placed on every bus in the network. In this experiment, 77 of 78 failed lines were correctly identified. In the remaining test case, the failed line was among the three with top scores.

In the experiment shown in Figure 5 PMUs were placed on buses 3, 12, and 33. Here, the effect of the error due to the linear approximation is increased by the low-rank projection $E$. In this case, 68 of 78 lines were correctly identified, a correct identification rate of 0.872. Even when the algorithm is incorrect, it gives the correct line one of the top 3 scores in every test. Furthermore, for all tests in which the algorithm mis-identified the failed line, the line that received that top score was incident to the correct line. That is, even when this algorithm is “wrong,” it often selects a line near the line that actually failed. We show an example in Figure 6.

In Figure 5 we show an experiment in which there is
Fig. 6: Line (24, 26) is the line removed in this test. Lines are colored and thickened according to $\sqrt{t_k}$. Line (26, 27) was chosen by the algorithm.

Fig. 7: Test of our algorithm on the IEEE 57-bus network with only a single PMU on bus 35. This represents a near-worst-case scenario. The algorithm still identified 56 of the 78 line failures correctly, with the correct line having a top 3 score in 72 of 78 line failures. In addition, in 12 of the 22 tests in which the algorithm chose incorrectly, the best-scoring line was incident to the correct line.

C. Linear Programming Extension

To test the accuracy of the linear programming approach in Equation (20), we conducted an experiment in which we removed every pair of lines in the IEEE 57-bus network and tried to identify the removed lines through the linear program. Those pairs that cause the power flow solver to not converge were not included. We show the results in Figures 8–10. Each column in the figures again represents a single experiment. For each experiment, the lines are ordered by the scores from the linear program, and the rankings of the two correct lines, $r_1$ and $r_2$, are recorded. The dashed blue line represents $\min(r_1, r_2)$.
in each experiment, while the solid red line represents $\max(r_1, r_2)$. The experiments are sorted by $\max(r_1, r_2)$. The linear program tends to give sparse solutions, so many lines are given a weight of 0. If a correct line is given a weight of 0, we assume it is ordered last, even though typically many lines have weight 0.

In our experiments, this formulation often identifies one failed line, and sometimes identifies both. In addition, it performs well with PMUs everywhere, suggesting that most of the error in the other cases is due to the low-rank projection. However, we need many more PMUs to identify all failed lines with this approach than we needed to identify one failed line with Algorithm [3].

V. Conclusion and Future Work

In this paper we have presented FLiER, a fast, practical algorithm to identify single-line topology errors based on a sparse deployment of PMUs. Though we have focused on identifying a line failure, our approach can be used for more general topology errors as well. Our algorithm requires that one obtain a state estimate shortly before the topology changes, and assumes that the network specifications remain unchanged or change in a known way as a result of the failure. Our method can be used in both distribution and transmission networks, and is compatible with “zooming in” on network switches, as many current topology error correction techniques do.

We have also extended this algorithm to a linear programming formulation that can identify multiple simultaneous line failures. To obtain good results with this algorithm, we required more PMUs than we needed with the single-line algorithm. In our simulations, the linear programming approach typically identified at least one line that failed, and frequently was able to identify more.

There are several remaining questions open for future work. We believe it is important to analyze the sensitivity of our technique to noise in PMU readings and errors in the pre-failure state estimate. We also believe it is possible to diagnose when the linear approximation is most likely to lead to incorrect diagnosis, and perhaps to do more computation to deal with those cases. In addition, we plan to extend our approach to handling other system events, such as single-phase line failures or changes in line parameters due to overloading.

REFERENCES

[1] A. Monticelli, “Electric power system state estimation,” Proceedings of the IEEE, vol. 88, no. 2, pp. 262–282, 2000.

[2] A. Ashok and M. Govindarasu, “Cyber attacks on power system state estimation through topology errors,” in Power and Energy Society General Meeting, 2012 IEEE, 2012, pp. 1–8.

[3] E. Caro, A. J. Conejo, and A. Abur, “Breaker status identification,” IEEE Transactions on Power Systems, vol. 25, no. 2, pp. 694–702, 2010.

[4] C. N. Lu, J. H. Teng, and W.-H. E. Liu, “Distribution system state estimation,” IEEE Transactions on Power Systems, vol. 10, no. 1, pp. 229–240, 1995.

[5] S. Naka, T. Genji, T. Yura, and Y. Fukuyama, “A hybrid particle swarm optimization for distribution state estimation,” IEEE Transactions on Power Systems, vol. 18, no. 1, pp. 60–68, 2003.

[6] T. Niknam and B. B. Firouzi, “A practical algorithm for distribution state estimation including renewable energy sources,” Renewable Energy, vol. 34, no. 11, pp. 2309–2316, 2009.

[7] A. Monticelli, “Modeling circuit breakers in weighted least squares state estimation,” IEEE Transactions on Power Systems, vol. 8, no. 3, pp. 1143–1149, 1993.

[8] K. A. Clements and A. S. Costa, “Topology error identification using normalized lagrange multipliers,” IEEE Transactions on Power Systems, vol. 13, no. 2, pp. 347–353, 1998.

[9] E. Lourenço, A. J. A. Costa, K. A. Clements, and R. A. Cernev, “A topology error identification method directly based on collinearity tests,” IEEE Transactions on Power Systems, vol. 21, no. 4, pp. 1920–1929, 2006.

[10] N. Vempati, C. Silva, O. Alsac, and B. Stott, “Topology estimation,” in Power Engineering Society General Meeting, 2005. IEEE, 2005, pp. 806–810.

[11] F. Vosgerau, I. S. Costa, K. A. Clements, and E. M. Lourenço, “Power system state and topology coestimation,” in Bulk Power System Dynamics and Control (iREP) - VIII (iREP), 2010 iREP Symposium, 2010, pp. 1–6.

[12] O. Alsac, N. Vempati, B. Stott, and A. Monticelli, “Generalized state estimation,” IEEE Transactions on Power Systems, vol. 13, no. 3, pp. 1069–1075, 1998.

[13] G. N. Korres and N. M. Manousakis, “A state estimation algorithm for monitoring topology changes in distribution systems,” in 2012 IEEE Power and Energy Society General Meeting, 2012, pp. 1–8.

[14] M. Zima-Bočkarjova, E. Scholtz, M. Larsson, and G. Andersson, “Critical consideration of the suitability of randomized optimization methods: Power system topology estimation problem,” in PowerTech, 2009 IEEE Bucharest, 2009, pp. 1–7.

[15] J. Krstulovic, V. Miranda, A. J. A. Simoes Costa, and J. Pereira, “Towards an auto-associative topology state estimator,” IEEE Transactions on Power Systems, vol. 28, no. 3, pp. 3311–3318, 2013.

[16] A. G. Phadke and J. S. Thorp, Synchronized Phasor Measurements and Their Applications. New York: Springer, 2008.

[17] J. Sexauer, P. Javanbakht, and S. Mohagheghi, “Phasor measurement units for the distribution grid: Necessity and benefits.” in 2013 Innovative Smart Grid Technologies, 2013, pp. 1–6.

[18] G. Sánchez-Ayala, J. R. Agüero, D. Elizondo, and M. Lelic, “Current trends on applications of pmus in distribution systems.” in 2013 Innovative Smart Grid Technologies, 2013, pp. 1–6.