A Systematic Approach of Identifying Optimal Load Control Actions for Arresting Cascading Failures in Power Systems

Ajay Chhokra  
Vanderbilt University  
Nashville, Tennessee  
chhokraad@isis.vanderbilt.edu

Amogh Kulkarni  
Vanderbilt University  
Nashville, Tennessee  
kulkaras@isis.vanderbilt.edu

Saqib Hasan  
Vanderbilt University  
Nashville, Tennessee  
saqibhasan@isis.vanderbilt.edu

Abhishek Dubey  
Vanderbilt University  
Nashville, Tennessee  
dabhishe@isis.vanderbilt.edu

Nagabhushan Mahadevan  
Vanderbilt University  
Nashville, Tennessee  
nag@isis.vanderbilt.edu

Gabor Karsai  
Vanderbilt University  
Nashville, Tennessee  
gabor@isis.vanderbilt.edu

1 INTRODUCTION

Emerging trends and challenges: Electrical power grids and operations are evolving in response to the growing volatility in their environments, requirements and increased emphasis on decentralized or distributed energy resources [16]. However, as these systems are evolving the existing legacy infrastructure is under pressure to improve reliability and reduce the likelihood of failures [11]. Intelligent software enabled protection equipment such as SEL-311C, GE-L90 [1, 2] are critical components of the modern electrical power grid. However, due to lack of system wide perspective, the actions of protection devices have been known to cause cascading failures leading to blackouts [5].

State of the art: Standard industry practice for identifying cascades is to perform on-demand simulations to predict the future state of the system in terms of imminent outages and find different corrective actions to stop the propagation. For example, [5] describes various cascade simulation models that can be used. However, performing simulations using these models is computationally expensive and cannot provide a quick real-time response. Once the cascade conditions have been identified, pre-defined actions such as load shedding can be used to suppress the cascading effects of overloads, voltage and frequency instabilities.

An alternative approach is to curtail a percentage of load instead of completely shedding it. For example, the cascading failures during the blackout of Aug 2003 in the USA could have been avoided by removing relatively small amount of load in the Cleveland area [6, 15]. Curtailment provides an effective means of handling the cascade effects without disconnecting the complete load. The most effective load curtailment is not always obvious. Line operators rely on optimal power flow algorithm to identify the suitable generator or load re-dispatch actions. A general practice is to use simple linear programming to find minimalistic load shedding actions that can prevent progression of cascade. But the linear approximation of underlying system can be misleading and can result in incorrect load management recommendations. A number of approaches based on model predictive control [3, 4, 15, 17, 20] have been proposed that tackles problems of voltage collapse and successive branch outages due to overloads. However, the model predictive control strategy is not always guaranteed to provide an optimal solution because of the limitation of underlying approximation of the mathematical model and limited number of control actions as per the control
rules amongst them. The second step uses the generated simulation model along with user supplied blackout criteria and cascade model to perform contingency analysis. The last step in the tool chain involves in identifying the loads that has to be curtailed and the amount of reduction for every contingency discovered in step 2. The load classification objective is achieved by performing sensitivity analysis to determine how a single load affects the branch flows. These steps are discussed in more detail in the following sections.

3 MODEL GENERATION

Power system models are large and complex that involve various kinds of components. Standard practice is to create these models by hand for different simulation platforms. A graphical environment with the capability of modeling power systems and thus generating different simulation models can be advantageous and time saving. The input to the model transformation process is the system specification file. The input file should clearly identify the system topology and component parameters. We are using IEEE Common Data Format to represent bus and branch data. Model Transformation process parses the CDF file and creates an intermediate graphical model.

The intermediate graphical model is based on domain specific language (DSML) developed in generic modeling environment (GME) [18]. The language allows the modeling of cyber-physical energy systems with both discrete (discontinuous) and continuous components along with their different kinds of interactions. The continuous components can be broadly classified as 1) Power Delivery Elements like transmission lines and transformers, 2) Power Conversion Elements like loads and generators, and 3) Interface Elements like buses. The discrete components include devices that supervise and control the state of continuous components like protection relays and breaker assemblies. The DSML provides an abstract interface of these components which can be extended by different parameters as per desired level of refinement. The intermediate model makes it easier for system designer to make changes in the system topology by adding or removing some components, changing their parameters or implementation details. For more information, please refer to [10] which explains the DSML in more detail and also lists the algorithm for generating Simulink models.

4 IDENTIFICATION OF CRITICAL CONTINGENCIES

A number of approaches have been suggested in the literature to identify critical contingencies, [9] uses a graph theoretic approach to obtain high order contingency lists which are otherwise infeasible by simple brute force method of randomly selecting components. [13] proposes a stochastic method based on random chemistry for listing important line outages. Various cascade simulation models like TRELSS [7], OAK [8], Manchester [19], DCSIMSEP [13] can be used to determine the progression of cascading failures.

For our study, we developed a simple cascade simulation model (based on steady state calculations) that successively solves the power flow (using OpenDSS) by removing the overloaded branches from the system after the initial component outages. The simulation keeps on tripping the overloaded branches till a blackout situation is reached or there are no more secondary effects (overloads) in the

2 OVERVIEW OF THE APPROACH

Figure 1 shows the overall structure of the tool chain. The load control strategies are obtained in three main sequential steps. In the first step a system specification in a standard IEEE common data format is translated to an intermediate graphical model which is then converted to different simulation models like OpenDSS, Simulink etc. The modeling language of the graphical model specifies the kind or types of components model used as well as connection horizon. Moreover, if there is a change in the system topology the optimization problem has to be formulated again.

Our Approach: In this paper we present a systematic approach with minimal human involvement that can help line operators for better understanding and thus handling emergent situations by 1) performing on demand simulations of different fidelity, steady state analysis (openDSS [12]) and transient analysis (Simulink/ Mathworks) 2) performing contingency analysis, 3) identifying optimal load control actions for arresting cascade progression, and 4) storing the load control actions for future lookup. Such a methodology can aid in making the power system more resilient as it can be part of the special protection scheme to prevent cascading line trippings leading to blackout. Our approach utilizes an open loop optimization method, based on OpenMDAO [14] (multi-disciplinary analysis and optimization framework) that uses external simulators (OpenDSS) for evaluating the mathematical model. A system designer can also benefit from this tool chain by exploring the design space by abstracting or refining the component models and or changing the power flow solver modes.

Paper outline: The main contribution of this paper lies in describing a tool chain that helps in 1) generating simulation models from system specification in IEEE Common Data Format, 2) identifying critical contingencies and 3) presenting minimal load curtailment strategies for the identified blackout causing contingencies in the form of a lookup table. The paper is organized as follows, section 2 gives an outline of the proposed approach followed by detailed description of model transformation in section 3, identification of critical component outages in section 4 and optimization problem in section 5. The section 6 describes the results of the different experiments followed by concluding remarks in section 7.

Figure 1: Complete tool chain for identifying load control actions to avoid system blackout
system. This cascade simulation model caters to slowly progressing cascades that eventually lead to blackouts involving overloads. We have adopted a conservative approach where all the secondary effects of initial outages are mitigated through the existing (pre-defined) protection schemes.

Algorithm 1 Algorithm for finding critical N-k contingencies

Input: Model, k, Branch
Output: T, TR
A ← choose( Branch, k)

for \( j \leq \frac{|\text{Branch}|}{k} \)
do
Prev ← A[ j ], Next ← \( \emptyset \), Temp ← \( \emptyset \), Start ← A[ j ]
Model.apply_contingency(Prev)
while True do
if Model.check_blackout() then
T ← T \( \cup \) Start
TR ← TR \( \cup \) Temp
BREAK
else
Next ← Model.get_overloads()
if Next \( \neq \emptyset \) then
Temp ← Temp \( \cup \) (Next \( \cup \) Prev)
Prev ← Temp
Model.trip_branches(Next)
else
BREAK
end if
end while
\( j \leftarrow j + 1 \)
end for

Algorithm 1 shows the underlying mechanism of finding N-k contingencies. The input parameter of the algorithm includes a OpenDSS model (Model), an integer representing the order of contingencies (k) and a set of all branch labels (Branch). The output of the algorithm is two sets \( T, TR \) that represents a collection of initiating events and their respective progressions. The set, \( T = \{s_1, s_2, \ldots, s_n\} \) is a collection of all contingencies that can cause blackout, where \( s_j \) is some combination of branch outages. The set, \( TR = \{(s_1, s_2), (s_2, s_3), \ldots (s_j, s_j)\} \) represents the progression of cascade caused due to \( s_j \), where \( s_j \) represents the initial branch outages and \( s_j \) implies the branch outages as a consequence of \( s_j \). The algorithm starts with tripping \( k \) lines at random and solving the power flow to update the branch currents and bus voltages. The second step is to check for the blackout criteria. In the current implementation, the blackout criteria is configurable in terms of the percentage of the original load (demand) that is still operational. In a given state, if more than 40% of the net system load demand cannot be satisfied, then the system is considered to have reached blackout.

If the system is not in a blackout state, the secondary effects of the branch outages are investigated by checking the overloads in rest of the system. If no overloads are found then, the system is considered to have reached a safe state from where it cannot reach blackout. On the other hand, if some secondary overloads were present, the transition relation, represented by Temp is updated followed by tripping all those branches. After branch tripping, the blackout criteria is checked again and the process repeats until a blackout state is reached or the system reaches a safe state where there are no overloads. The loop terminates when the cascade progression reaches a safe state (no further cascades) or the blackout criteria is met.

The simulation model supports the following operations:

- **check_blackout()**: This function determines if the system, in its current state, satisfies the blackout constraint. The method returns True if the system fulfills blackout criteria, and False otherwise.
- **get_overloads()**: This function returns the labels of all branches which are overloaded. **trip_branches()**: This function modifies the topology of the system by branch tripping specified by the parameter. **apply_contingency(labels)**: This function changes the topology of the system by removing the branches listed in the set labels.

5 OPTIMIZATION FRAMEWORK

Most of the cascading failures in power system have been caused by the protection elements associated with transmission lines, transformers and buses reacting to high currents and low voltages. An overload in transmission line can persist till the conductor sags and touches the ground or the over current protection kicks in to isolate the line. A severe overload can also be mistreated by a distance relay to be a distant zone 3 fault causing a zone 3 relay to trip the line in 1-2 seconds. If these secondary effects of increase in branch current and decrease in bus voltage can be corrected then the cascade can be arrested through load and generator control actions.

This problem of finding such control actions can be formulated as a non-linear programming problem using steady state power flow solver as shown below.

\[
\text{minimize} \quad \text{Cost}(\Delta L, \Delta G)
\]

subject to:
\[
\forall k_l, \quad 0 \leq |k_l| \leq K_l^{\text{Max}}, \quad k, l \in \mathbb{Z}, k \neq l
\]
\[
\forall j, \quad V_j^{\text{Min}} \leq V_j \leq V_j^{\text{Max}}, \quad j \in \mathbb{Z}
\]
\[
\forall m, \quad 0 \leq |\Delta L_m| \leq L_m^{\text{Max}}, \quad m \in \mathbb{Z}
\]
\[
\forall n, \quad G_n^{\text{Min}} \leq |\Delta G_n| \leq G_n^{\text{Max}}, \quad n \in \mathbb{Z}
\]
\[
\forall i, \quad \sum_m |\Delta L_i| \leq L_i^{\text{Max}} \quad i \in \mathbb{Z}
\]
\[
\forall m, \quad \frac{\text{Im}(L_m - \Delta L_m)}{\text{Re}(L_m - \Delta L_m)} = \frac{\text{Im}(L_m)}{\text{Re}(L_m)}, \quad m \in \mathbb{Z}
\]
\[
V, I = f(L + \Delta L, G + \Delta G)
\]

where, \( L, G \) are vectors consisting of pre-control complex load powers and generator power injections respectively. \( \Delta L_p, \Delta G_q \) are the changes in the load demand \( L_p \) and generator injection \( G_q \). \( L_p^{\text{Max}} \) is the maximum absolute load demand related to load \( L_p, G_q^{\text{Min}/\text{Max}} \) are the minimum and maximum absolute value of power injected by generator \( G_q \). \( I \) is a two dimensional vector of

\( \text{choose}(A) \) returns all combinations of the members of \( A \) i.e. power sets. Hence \( A \) consists of list of all possible initial outages that have to be examined. **Temp** is an array of tuples that temporarily stores the evolution of an initial outage.
complex branch currents, where $I_n$ implies all the currents injected at bus $n$ (cumulative sum of all the elements in column $n$) and $I_{kl}$ represents the current flowing along the branch between node $k$ and $l$. The scalar $I_{kl}^{Max}$ is the maximum permissible amount of current (absolute) that can flow along the branch $kl$. $V_n$ is a vector of complex bus voltage and $V_n^{Min/Max}$ represent minimum and maximum absolute value of bus voltage. The function, $f(L + \Delta L, G + \Delta G)$ represents the power flow solver which updates the current and voltage vectors as per the current load and generation profile identified by vectors $L$ and $G$.

The objective function $Cost(\Delta L, \Delta G)$ in (1) represents the social costs incurred during load curtailment and rapid change in generation cycles. The function can be modified to reflect the critical loads representing hospitals and government establishments by assigning appropriate weights. The function also includes the financial burden caused due to emergent reduction in power generation that can cause equipment damage resulting from sudden deceleration. The inequality constraint (2-3) identify the branch current and bus voltage violations. These constraints enforce voltage stability and no overloaded branch is present in the final solution, if the optimization problem converges. These inequalities can be extended to ensure system frequency stability and generator out of phase security. The inequality constraints (4-5) describes the upper bound on the net load that can cause equipment damage resulting from sudden deceleration.

The number of iterations required by an optimization algorithm depends upon the number of design variables i.e input state space. The optimization problem can converge in less time (if there exists a solution) if the input state space is smaller i.e. by removing those loads that do not affect branch overloads as identified in the cascade progression of a given contingency. Moreover, it is well known that the optimization results are sensitive to the initial estimates of the design variables. The following subsection discusses the sensitivity analysis implemented as a part of the tool chain.

Sensitivity Analysis - Figure 2 shows the implemented sensitivity analysis procedure. The process can be broken into following three sequential steps:

1. **Data point generation:** In this step, the effect of a varying absolute value of load demand at a given bus on all the branch currents is observed. The load ($L_p$) vs branch current ($I_{kl}$) data points are stored for all system loads. We have used Full Factorial Design of Experiment (DoE) analysis that uniformly samples the input space i.e. range $[0, L_p^{Max}]$ for each load $L_p$ in system. In our implementation 100 data points are considered for each load.

2. **Regression Analysis:** This step involves finding equation parameters (slope and intercept) for branch current vs load change data points generated in the previous step. It is safe to assume linear relationship between branch currents and load demand since the power factor remains constant. The sensitive loads can be classified by observing the slope of the equation. $I_{kl} = mL_p + c$, where $I_{kl}$ and $L_p$ are the absolute value of load $p$ and the branch current (absolute) between nodes $k$ and $l$; and $m, c$ are the equation parameters.
Table 1: Power Flow Summary for IEEE 14 Bus system

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Per unit voltage (Max, Min)</td>
<td>1, 0.9463</td>
</tr>
<tr>
<td>Total Generation (Active, Reactive)</td>
<td>261.139 MW, 80.0561 Mvar</td>
</tr>
<tr>
<td>Total Load (Active, Reactive)</td>
<td>259 MW, 73 MVar</td>
</tr>
<tr>
<td>Total Losses (Active, Reactive)</td>
<td>1.96291 MW, 6.49788 Mvar</td>
</tr>
</tbody>
</table>

Table 2: OpenMDAO parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem type</td>
<td>Non Linear</td>
</tr>
<tr>
<td>Solver</td>
<td>Sequential Least Squares Programming</td>
</tr>
<tr>
<td>Derivative Calculation</td>
<td>Forward finite difference</td>
</tr>
<tr>
<td>Step Size</td>
<td>2500.00</td>
</tr>
<tr>
<td>Max Iterations</td>
<td>1000</td>
</tr>
</tbody>
</table>

For a given load, if the slope is positive for any branch current, then the load is considered to be sensitive.

Starting Point Estimation: The starting value estimate is calculated in this step for all the sensitive loads identified in the previous step by solving the linear equation, \( \frac{I_{kl}^{Max}}{m_L} = m_L p + C \), for \( L_p \). The staring value is equal to \( a = \frac{k_{kl}^{Max} - C}{m_L} \), such that \( 0 \leq \frac{k_{kl}^{Max} - C}{m_L} \leq L_p^{Max} \) otherwise \( a = 0 \).

6 RESULTS

In order to validate the accuracy of the generated load curtailment actions, IEEE 14 bus system is used, as shown in Figure 3. The IEEE 14 bus system is modified by replacing all generators and condensers with appropriately rated voltage sources as we are using a steady state power flow solver. Table 1 summarizes the solved power flow for IEEE 14 bus system in OpenDSS. We identified 441 cases of initial critical branch outages that satisfies the blackout criteria, based upon our conservative cascade simulation model defined in section 4. Its important to highlight here that this number will change if some other blackout criteria or cascade simulation model is used. Out of 441 cases, the optimization routine was able to find a solution in 427 cases with an average of 29 iterations. The corresponding load control actions are stored in the form of hash table. Figure 4 shows the % load demand reduction in 427 cases. In all solved cases, load curtailment is restricted to less than 20 % of the net system load (constraint 6) with an average load reduction of approximately 8%. In the 14 unsolved cases, the optimization problem could not converge to an optimal solution due to 1) maximum iteration limit (1000) reached 2) solver is stuck close to saddle point. Table 2 lists the optimization framework parameters used for the experiments.

Figure 3 also shows the evolution of one of the contingencies evaluated. The green region in the figure implies the first stage overloads while the purple region shows the effects of initial and first stage branch outages. A 3 phase to ground phase fault is injected in line, \( L_4,5 \). The fault is isolated by tripping the line which leads to overloading of lines \( L_3,4, L_2,4 \) and \( L_3,2 \). These overloads

are removed by tripping these lines as per pre-defined protection schemes. The removal of these secondary effects leads to overloads in lines \( L_6,11, L_13,14, L_9,10, L_4,9 \). The removal of these overloaded branches matches the blackout criteria (as defined in the blackout criteria mentioned in 4). In order to arrest the cascade progression, some load curtailment has to be done in order to remove overloading in branches \( L_3,4, L_2,4 \) and \( L_2,3 \). As per the generated look up table, load demand at Bus 4, labeled as load 10, has to be reduced by 6% to arrest the cascade progression.

Figure 4: The figure on the top shows the net load loss (percentage) and the bottom one shows the number of iterations taken by the optimization engine to find a solution.
Scalability Analysis: Since power systems are large networks its imperative to discuss the impact of scale on our approach. The presented workflow consists of 3 major computational tasks 1) Model Transformation, the run time complexity of this routine increases linearly with the increase in components in power network as its a bitwise transformation from one class to another. 2) N-k Contingency Analysis, For small values of k (1, 2, 3 in our case), this process has approximately polynomial run time complexity as the number of combinations increases polynomially as well as the time required for solving power flow (DC). 3) Optimization, The number of constraints (one per branch) and design or control variables (loads) increases linearly with the increase in the size of the power network. It is well known that the performance of the optimizer is greatly affected by starting point estimate and the size of the input space of the problem. The sensitivity analysis routine uses full factorial based analysis to estimate a starting point for each load to prevent cascade. For large systems the number of sensitive loads might be very large. A bound on number of control variables (loads) can be placed that can reduce the search space for the optimization problem. Figure 5 shows the total load percentage reduction and number of iterations if two of the most sensitive loads are selected (Average number of control variables in first experiment was 7). The average number of iteration have reduced from 29 to 25 with a slight increase in percentage load reduction. The total time taken by the optimization problem has also decreased from 155 minutes to 120 minutes. However, the number of solved cases remained same in both the experiments.

7 CONCLUSION

In this paper we presented a complete workflow that generates simulation models, identifies critical blackout causing contingencies and utilizes extensible optimization methodology based on OpenMDAO and OpenDSS to identify load control actions to avoid cascading outages. We validated our approach by using a IEEE 14 bus system and showed the load control actions associated with one of the identified contingencies. We wish to extend our blackout criteria to include voltage and frequency stability and extend the

OpenMDAO interface to integrate with dynamic simulation models to capture transient effects.

ACKNOWLEDGMENTS

This work is funded in part by the National Science Foundation under the award number CNS-1329803. The authors will like to thank Rishabh Jain and Dr. Srdjan Lukic from North Carolina State University for their help and discussions related to the work presented here.

REFERENCES