Power System Security Enhancement for Real-Time Operations

During Multiple Outages using Network Science

by

Reetam Sen Biswas

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved April 2021 by the
Graduate Supervisory Committee:

Anamitra Pal, Chair
Vijay Vittal
John Undrill
Meng Wu
Yingchen Zhang

ARIZONA STATE UNIVERSITY

May 2021

ABSTRACT

Ensuring reliable operation of large power systems subjected to multiple outages is a challenging task because of the combinatorial nature of the problem. Traditional methods of steady-state security assessment in power systems involve contingency analysis based on AC or DC power flows. However, power flow based contingency analysis is not fast enough to evaluate all contingencies for real-time operations. Therefore, real-time contingency analysis (RTCA) only evaluates a subset of the contingencies (called the contingency list), and hence might miss critical contingencies that lead to cascading failures.

This dissertation proposes a new graph-theoretic approach, called the feasibility test (FT) algorithm, for analyzing whether a contingency will create a saturated or overloaded cut-set in a meshed power network; a cut-set denotes a set of lines which if tripped separates the network into two disjoint islands. A novel feature of the proposed approach is that it *lowers the solution time* significantly making the approach viable for an exhaustive real-time evaluation of the system. Detecting saturated cut-sets in the power system is important because they represent the vulnerable bottlenecks in the network. The robustness of the FT algorithm is demonstrated on a 17,000+ bus model of the Western Interconnection (WI).

Following the detection of post-contingency cut-set saturation, a two-component methodology is proposed to enhance the reliability of large power systems during a series of outages. The first component combines the proposed FT algorithm with RTCA to create an integrated corrective action (iCA), whose goal is to secure the power system against post-contingency cut-set saturation as well as critical branch overloads. The second

component only employs the results of the FT to create a relaxed corrective action (rCA) that quickly secures the system against saturated cut-sets.

The first component is more comprehensive than the second, but the latter is computationally more efficient. The effectiveness of the two components is evaluated based upon the number of cascade triggering contingencies alleviated, and the computation time. Analysis of different case-studies on the IEEE 118-bus and 2000-bus synthetic Texas systems indicate that the proposed two-component methodology enhances the scope and speed of power system security assessment during multiple outages.

This work is dedicated to

my loving and caring parents (Sumana Biswas and Tapas Kumar Sen)

and maternal grandparents (Bithika Biswas and Ashish Baran Biswas).

ACKNOWLEDGMENTS

I express by sincerest gratitude to my advisor Dr. Anamitra Pal, for his support and guidance throughout the course of this research work. Pursuing doctoral research has always been my long-cherished dream, and I am thankful to him for giving me this opportunity at the Arizona State University. During the years working with him, I have gained extensive research experience and broadened my horizon in the power systems area.

I am grateful to Dr. Vijay Vittal for his help and support throughout my research. I am indebted to Dr. John Undrill, for his continuous enthusiasm in my work. Dr. Undrill's insightful comments and constructive criticism contributed significantly to the betterment of my research. I am grateful to Dr. Meng Wu and Dr. Yingchen Zhang for their genuine interest in my work and providing their invaluable feedback during my Comprehensive exam. I would also like to thank Dr. Yingchen Zhang for providing me the opportunity to do exciting summer internships at the National Renewable Energy Laboratory (NREL), Golden, Colorado. My experience at NREL has been enriching.

I convey a very special "thank you" to Dr. Trevor Werho. He has always extended a helping hand whenever I needed it throughout the course of this research. I am also thankful to Power Systems Engineering Research Center (PSERC) for funding this work through PSERC grants S-74 and S-87.

Last, but not the least, I express my deepest gratitude towards my family members for their unconditional love, support, and motivation during my doctoral research. Their unwavering faith has been crucial in the successful completion of my dissertation.

TABLE OF CONTENTS

Page

CHAPTER Page

CHAPTER Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

NOMENCLATURE

c_l^{FT}	Directed weight associated with branch e_l from bus v_l^F towards bus v_l^T in the latent capacity graph ($\mathcal{C}$).
c_l^{TF}	Directed weight associated with branch e_l from bus v_l^T towards bus v_l^F in the latent capacity graph ($\mathcal{C}$).
C_1	The set of buses contained in cluster 1.
C_2	The set of buses contained in cluster 2.
$C_{\mathcal{P}}$	Maximum extra flow that can be transferred along path $\mathcal{P}$ from a source towards a sink.
CA	Contingency Analysis
$\mathcal{C}$	A bi-directional latent capacity graph of the power network.
e_l	l^{th} branch in set E.
E	A set containing all branches of the power network.
E_v	A set containing critical branch contingencies, detected by RTCA that result in post-contingency branch overloads.
E_s	A set containing the special assets detected by the Feasibility Test (FT) algorithm.
Δf_l	Change in power flow on the l^{th} branch.
Δf	Total power transferred from one specific bus (i) to another specific bus (j).
f_l	A flow associated with branch e_l.
f_l'	New flow corresponding to the changes in bus power injections in the system.

f_k^0	Original flow on the k^{th} branch before it was outaged.
f_l^c	Post-contingency flow on the l^{th} branch with the k^{th} branch out.
f_l^{max}	Rating of branch e_l.
F	The flow f_l through branch e_l is assigned to variable F in the UPS algorithm to update graph-theory based network flows for the outage of branch e_l.
$F_{\mathcal{P}}$	The flow injected along path $\mathcal{P}$.
FERC	Federal Electricity Regulatory Commission
FT	Feasibility Test.
$\mathcal{F}$	A directional flow graph of the power network.
g	The total number of generator buses in the system.
G	A set containing the locations of generator buses.
G_i	Active power generated at generator bus $i \in G$.
ΔG_i	Change in active power generation at the generator bus $i \in G$.
$\mathcal{G}$	An undirected weighted graph of the power network.
ΔI^p	Net increase in power injection at a bus $v_p \in V^p$.
ΔI^n	Net decrease in power injection at a bus $v_n \in V^n$.
iCA	Integrated Corrective Action.
J	Jacobian matrix obtained for an AC power flow solution.
k	Total number of branches in cut-set K.
K	Any cut-set in the power network.
K_{crit}	Limiting critical cut-set for branch $e_l \in E$.
$\mathcal{K}_{crit}$	A set containing all the limiting critical cut-sets detected by FT.

KCL	Kirchhoff's Current Law
KVL	Kirchhoff's Voltage Law
L	A set containing all the load buses.
L_j	Active power demand at a bus j in set L.
ΔL_j	Change in active power demand at bus j in set L.
$LODF_{l,k}$	Line Outage Distribution Factor for branch e_l corresponding to the outage of branch e_k in the system.
$LOIF_k$	Line Outage Impact Factor for a branch contingency e_k.
$M-UPS$	Modified Update Scheme.
$M-SA$	Modified Shortlisting Assets.
n	Total number of buses in a system.
NERC	North American Electric Reliability Corporation
p_j	Active power injection at the bus j in the system.
Δp	This set contains the active power mismatches for different buses in the iterations of an AC power flow solution.
$p_{j,k}$	Active power flowing from bus j towards the bus k through the corresponding branch.
P_G^1	Total active power generation in cluster C_1.
P_G^2	Total power generation in cluster C_2.
P_L^1	Total active power demand in cluster C_1.
P_L^2	Total power demand in cluster C_2.
ΔP^1	Net active power injection in cluster C_1.

ΔP^2	Net active power injection in cluster C_2.
P_K	Total active power to be transferred across cut-set K.
$\mathcal{P}$	This is a path containing a sequence of branches from a source bus to a sink bus in a connected graph.
$PTDF_{l,i}^{j}$	Power Transfer Distribution Factor for the l^{th} branch when power is added at the bus i and withdrawn at the bus j.
q_j	Reactive power injection at the bus j in the power system.
Δq	This set contains the reactive power mismatches for different buses in the iterations of an AC power flow solution.
rCA	Relaxed Corrective Action
R_K	Total active power transfer capacity of cut-set K.
RTCA	Real Time Contingency Analysis (RTCA)
SA	Shortlisting Assets.
SCED	Security Constrained Economic Dispatch
T_l^i	Transfer margin of the i^{th} saturated cut-set, associated with branch e_l.
T_l	Transfer margin of the limiting critical cut-set associated with branch e_l.
$\mathcal{T}_k$	Electrical betweenness for a potential branch contingency e_k.
$\mathcal{T}_k^p$	Positive electrical betweenness for a potential branch contingency e_k.
$\mathcal{T}_k^n$	Negative electrical betweenness for a potential branch contingency e_k.
TC_l	Total additional active power transfer capability of the indirect paths of branch e_l.
UPS	Update Scheme.

v_l^F	The "from bus" of branch e_l.
v_l^T	The "to bus" of branch e_l.
V	A set containing different buses of the power system.
V^p	A set containing buses where the net power injection has increased after the corrective action, with respect to the original test case.
V^n	A set containing buses where the net power injection has decreased after the corrective action, with respect to the original test case.
$\mathcal{V}_j$	Voltage magnitude at bus j.
$\Delta\mathcal{V}$	A set containing the changes in bus voltage magnitude in different iterations of an AC power flow solution.
x	Total number of cut-sets associated with branch e_l.
χ_{jk}	Reactance of the branch joining bus j to bus k
y	Total number of saturated cut-sets associated with branch e_l.
Y	The bus admittance matrix of the network.
Y_{jk}^r	Real component of the bus admittance corresponding to the j^{th} row and k^{th} column of the Y matrix
Y_{jk}^i	Imaginary component of the bus admittance corresponding to the j^{th} row and k^{th} column of the Y matrix
z	A variable denoting impedance of a branch.
θ_j	Voltage angle at the bus j.
$\Delta\theta$	A set containing the change in bus voltage angles in different iterations of an AC power flow solution.

CHAPTER 1

INTRODUCTION

This Chapter presents the background and motivation for this research, followed by a detailed literature survey, and the research scope for this dissertation.

1.1 Background

Maintaining un-interrupted supply of electricity is of paramount importance, to satisfy the ever-increasing energy demands of the society. Failure of any element may have a negative impact on the normal operations of electric power systems. Real-time system monitoring is the first step to operate power systems reliably. Measurements are collected from the remote terminal units (RTUs) or local control centers, which are used to perform state estimation to determine the real-time status of the power system defined by the voltage magnitude and voltage angle of all buses in the system [1]-[2].

Phasor measurement units (PMUs) or synchrophasors built in the 1980s have improved the real-time monitoring capabilities of modern power systems significantly [3]-[4]. PMUs provide fast time synchronized measurements (typically 30 samples per second [5]-[6]. Fast reporting rates of voltage and current phasor measurements from PMUs facilitate quick, reliable state estimation and system monitoring in power transmission and distribution systems [7]-[16]. Due to the high cost associated with the synchrophasor technology significant research has been done on optimal PMU placement techniques to minimize the number of PMUs required for accurate state estimation [17]-[24]. Going beyond the advancements made in real-time state estimation and system monitoring, it is important to

enhance the state-of-the-art techniques of power system security assessment, which utilizes the converged state estimation results to investigate the consequence of potential contingencies in power systems.

The North American Electric Reliability Corporation (NERC) recommends that a reliable electric grid should be able to withstand the loss of a single element of its bulk power system (called *N*-1 reliability) [25]. Consequently, power system operators perform real-time contingency analysis (RTCA) and security constrained economic dispatch (SCED) successively at regular intervals [26]. RTCA evaluates the impact of a potential contingency on the system's static security (branch overloads and voltage violations). The critical contingencies detected by RTCA are modeled as the security constraints in SCED to provide a least-cost dispatch solution to eliminate the potential post-contingency overloads [27]-[28]. Despite RTCA-and-SCED trying to ensure *N*-1 reliability, cascading failures do occur in a power system. Severe instances of cascading failures could result in unintentional islanding and consequently, blackouts/brownouts [29]-[30]. Few examples of the major system disturbances in North America are as follows: Northeast Blackout in 1965 [31], New York City blackout in 1977 [32], Western Electricity Coordinating Council (WECC) blackout in 1996 [33], North East blackout in 2003 [34], power outages in Louisiana during Hurricane Gustav in 2008 [35], the US Southwest blackout in 2011 [36], and large-scale power interruptions in Florida during Hurricane Irma in 2017 [37].

Analysis of some of the major blackouts that have happened in the past has indicated that they often involve successive outages of power system assets [38]. For example, the 1977 New York City blackout was caused by the loss of 11 transmission lines in 52 minutes [32]. The Federal Electricity Regulatory Commission (FERC) reported that one of

the causes of the blackout was "the failure to recognize that a critical interconnection to the West was effectively unavailable" [39]. Werho et al. stated that a critical interconnection does not necessarily refer to a single line whose status can be monitored [40]; i.e., *a critical interconnection might consist of multiple transmission lines*. As such, real-time detection of critical interconnections in the power system that is suffering from multiple outages is a challenging task [41]. Moreover, considering the high speed with which some of the black-outs/brownouts occur (the 2011 US Southwest blackout occurred within 11 minutes [36]), it is clear that *fast* and *robust* assessment of power system security is extremely important for real-time operations.

The traditional approaches for providing situational awareness during real-time operations are based on steady-state contingency analyses techniques that solve AC or DC power flows [42]-[46]. However, power flow-based contingency analysis (CA) is not fast enough to perform an *exhaustive N*-1 RTCA [46]. The computational burden of the problem increases further for a more ambitious *N*-k contingency analysis [42]. Therefore, power utilities select a *subset* of the contingencies for evaluation based on some pre-defined criteria [44]-[45]. In [46], Huang et al. stated that the size of this subset has considerable impact on RTCA solution: a *large subset is computationally burdensome, while a small subset might miss critical scenarios*. This can be a problem for real-time operations during extreme scenarios when multiple outages occur in rapid succession [40].

Furthermore, transmission system operators do not necessarily monitor their neighboring systems in detail during state estimation or contingency analysis [40]. As a result, situational awareness could be limited if all external contingencies are not evaluated by RTCA. The US Southwest blackout of September 8, 2011 is a classic example of the

dangers that lack of situational awareness of critical events occurring in the neighboring system, pose [36]. The event was initiated by the loss of a 500-kV line (Hassayampa-North Gila) that was transporting power from Arizona to California through the Imperial Irrigation District (IID) system. This event occurred at 3:27 PM. Two transformers in the IID system (Coachella Valley and Ramon transformers) were overloaded and tripped offline. The Coachella Valley transformers tripped at 3:28 PM, while the Ramon transformers tripped at 3:32 PM. Following these initial triggering outages, a sequence of outages followed in the next 11 minutes disconnecting the San Diego area (which was being supplied power by the San Diego Gas & Electric (SDG&E) system) from the rest of the grid by 3:38 PM. Subsequent analysis of the blackout revealed that the neighboring entities had partial visibility of the IID's system, and as such could not observe that the Coachella Valley and Ramon transformers would be overloaded for the outage of Hassayampa-North Gila transmission line. Therefore, there is a genuine need to improve upon the existing methods of power system security assessment by enhancing situational awareness of critical contingencies in large power systems.

1.2 Literature Survey

1.2.1 Power Flow Studies

A power flow study forms the basis of steady-state analysis in an interconnected power system. The transmission networks are normally assumed to be three-phase balanced, and hence only the positive sequence network is modeled. The most commonly used power flow models used in the literature are AC power flow model, and the simplified DC power flow model [47].

AC power flow model is a non-linear model because the bus power injection is a function of the square of the bus voltages. The objective of the AC power flow problem is to obtain complete voltage magnitude and angle information for every bus of the power system for specified load and generation values. Depending upon the bus type, different buses in the power system are associated with known and unknown quantities. The three basic bus types are the PV, PQ, and slack bus.

A PV bus is a generator bus; the active power and voltage magnitude is known at the generator bus. A PQ bus is a load bus; active power and reactive power at all PQ buses are known. The slack bus (also referred as the swing bus) is a generator bus which has a large amount of generation capacity. The voltage magnitude and angle information at the slack bus is known. For the PV buses, the generator reactive power output adjusts automatically to maintain the specified voltage. However, reactive power capability of a generator is associated with a minimum and maximum limit. Therefore, a PV bus might switch to a PQ bus when the reactive power of the generator reaches its limit.

Now, the non-linear power flow equations are given as follows:

$$p_j = \sum_{k=1}^{n} \mathcal{V}_j \mathcal{V}_k \left(Y_{jk}^r cos\theta_{jk} + Y_{jk}^i sin\theta_{jk} \right) \tag{1.1}$$

$$q_j = \sum_{k=1}^{n} \mathcal{V}_j \mathcal{V}_k \left(Y_{jk}^r cos\theta_{jk} + Y_{jk}^i sin\theta_{jk} \right) \tag{1.2}$$

where, p_j and q_j are the net active and reactive power injected at bus j respectively, Y_{jk}^r and Y_{jk}^i denote real and imaginary parts of the bus admittance matrix Y respectively, corresponding to the j^{th} row and the k^{th} column, and θ_{jk} denotes the difference between the voltage angles between the buses j and k; i.e., $\theta_{jk} = \theta_j - \theta_k$. Considering that a system

has n buses and g generator buses, there are $2(n-1)-(g-1)$ unknowns in the system [48]. This is because the voltage magnitude for all generator buses are known, and the voltage magnitude and angle of the slack bus is also known.

There are different methods for solving the set of non-linear equations described by (1.1) and (1.2). The most commonly used method is the Newton Raphson method. It starts with an initial guess of the unknown variables, following which a Taylor series approximation is used to linearize the system at the given operating point, which can be expressed as follows:

$$\begin{bmatrix}\Delta\theta\\ \Delta\mathcal{V}\end{bmatrix} = -J^{-1}\begin{bmatrix}\Delta p\\ \Delta q\end{bmatrix} \tag{1.3}$$

where, Δp and Δq contain active and reactive power mismatches for different bus, and J is the Jacobian matrix obtained from the partial derivatives: $J = \begin{bmatrix}\frac{\partial \Delta p}{\partial \theta} & \frac{\partial \Delta q}{\partial \mathcal{V}}\\ \frac{\partial \Delta q}{\partial \theta} & \frac{\partial \Delta q}{\partial \mathcal{V}}\end{bmatrix}$. The linearized equations (1.3) are solved to obtain the next guess of the unknown variables iteratively. The iterations are repeated unless the mismatch in Δp and Δq is less than a specific tolerance.

The Newton Raphson method of solving the power flow equations is computationally expensive, because of the detailed network model, and due to the issue of the Jacobian being re-computed in each iteration based upon the partial derivatives. Fast decoupled power flow is a variation of the Newton-Raphson that exploits the approximate decoupling of active and reactive power flows in power networks, and moreover fixes the value of the Jacobian during different iterations to improve the computational efficiency [49].

To enhance the computational speed of the network analysis further, a DC power flow model is often used. Especially when reactive power and voltage magnitude are not of major concern, an approximate DC model can be used for solving the power flow problem. The network conductance is assumed to be zero (considered negligible in comparison with the reactance); i.e., the transmission line losses are ignored [47], [48]. Then the simplified transmission line power flow is given by:

$$p_{j,k} = \frac{\theta_j - \theta_k}{\chi_{jk}} \tag{1.4}$$

where, $p_{j,k}$ denotes the active power flow from bus j towards bus k, χ_{jk} denotes the transmission line reactance, and θ_j, θ_k denote the respective bus voltage angles. The approximate DC model helps to avoid the non-linearity of the AC power flow model. Therefore, information regarding the active power and voltage angle could be easily obtained using the DC power flow.

1.2.2 Contingency Analysis (CA)

Contingency analysis (CA) is a "what if" scenario simulator that evaluates the impact of an unplanned outage on the electric power system [50]. A contingency denotes the loss of a failure of a component of the power system. Generator contingency denotes the outage of a generator. Transmission contingency refers to the outage of a transmission line or a transformer. When generation is lost, much of the deficient power comes from tie lines, and this can mean line flow limit or bus voltage limit violations [47]. When a transmission line or a transformer fails, the flow on that line goes to zero and all flows nearby will be affected, which might result in a line flow limit and bus voltage limit violation.

Transmission contingencies are more common than generation contingencies [51]. This dissertation only relates to transmission contingencies occurring in a power system.

Contingency analysis can be conducted in day-ahead or real-time [51]. Day-ahead contingency analysis evaluates the effect of contingency on system reliability and identifies the active network constraints for day-ahead scheduling. RTCA identifies the consequence of contingencies that might occur in a very short time. RTCA helps operators to react quickly to unexpected outages. This dissertation only relates to real-time power system operations.

Contingencies could either be a single element contingency or a multi-element contingency [51]. A single and a multi-element contingency is denoted by *N*-1 and *N*-k, respectively. Contingency analysis has been traditionally limited to *N*-1 due to computational burden. For every potential contingency a power flow simulation is performed to evaluate the impact of the contingency. For very large power systems, the traditional RTCA-SCED framework is not able to perform an exhaustive *N*-1 evaluation within a few minutes [27]. In practice, only a subset of the potential contingencies is fed as input to RTCA; these selected contingencies form the contingency list [46]. The contingency list is determined from offline studies [27]-[28], operator knowledge [26], [50], or contingency ranking techniques [52]-[56]. Different contingencies in the contingency list are evaluated sequentially by a power flow solution. The AC power flow solution checks for both post-contingency branch overloads and bus voltage violations, while a DC power flow checks only for post-contingency branch overloads.

1.2.3 Linear Sensitivity Distribution Factors

Evaluating thousands of possible outages becomes a challenging problem to solve, if the results are to be presented within a few minutes [46]. One of the easiest ways to present a quick indication of possible overloads is to use the *linear sensitivity factors*. These sensitivity factors detect approximate changes in branch flows for generation changes or branch outages in the network and originate from the simplified DC power flow model. The two sensitivity factors commonly used in power system operations are [47], [53]-[57]:

1. Power Transfer Distribution Factor (PTDF)
2. Line Outage Distribution Factor (LODF)

The PTDF represents the sensitivity of the flow on branch e_l to a shift of power made from the bus i to bus j [47]. The PTDF is defined as follows:

$$PTDF_{l,i}^{j} = \frac{\Delta f_l}{\Delta f} \tag{1.5}$$

where, $l =$ branch index, $\Delta f_l =$ change in power flow on the l^{th} branch, and $\Delta f =$ total power transferred from the bus i to the bus j.

If ΔG_i denotes the power which is injected at bus i and withdrawn at the reference bus r, the updated branch flow is obtained as follows:

$$f_l' = f_l^0 + PTDF_{l,i}^{r} \Delta G_i \tag{1.6}$$

where, r denotes the location of the reference bus, f_l^0 denotes the previous flow, and f_l' denote the new flow after change in bus power injections.

The LODF describes the redistribution of branch flows due to a branch outage. The LODF is mathematically defined as follows [47]:

$$LODF_{l,k} = \frac{\Delta f_l}{f_k^0} \tag{1.7}$$

where, $LODF_{l,k}$ = Line outage distribution factor for the l^{th} branch after the outage of the k^{th} branch in the system, Δf_l is the change in flow in the l^{th} branch, and f_k^0 is the original flow in the k^{th} branch before it suffered an outage. Therefore, the post-contingency branch flow on l^{th} branch for an outage of the k^{th} branch is given as follows:

$$f_l^c = f_l^0 + LODF_{l,k} f_k^0 \tag{1.8}$$

where, f_l^0, f_k^0 denote pre-outage flows on branches l and k respectively, and f_l^c is the post-contingency flow on l^{th} branch with the k^{th} branch out.

PTDFs and transmission line ratings were used for screening out critical contingencies in [53]-[54], while LODFs were used for contingency screening in [55]. LODFs have also been used for quickly detecting an island formation due to multiple element contingencies [56]. A closed form expression of generalized LODFs under multiple line outages was presented in [57]. In [58], the PTDFs and generalized LODFs were used to detect island formation in power systems under multiple line outages. In [59], a dual computationally efficient method for calculating the PTDFs was proposed. In [60], contingency screening was done using LODFs. A variation of a DC power flow based linear sensitivity analysis was used to detect an island formation due to a potential contingency in [61].

1.2.4 Mitigation of Post-contingency Violations

Contingencies that result in post-contingency violations with regards to static security (branch overloads and voltage violations) are called *critical contingencies*. The system must be pre-positioned via appropriate actions to mitigate the impact of the critical

contingencies [51]. Otherwise, a cascading failure might be triggered by such contingencies resulting in an unforeseen blackout. The commonly used approaches to handle post-contingency violations are security constrained economic dispatch (SCED) [62], transformer tap adjustment [63], phase-shifter angle adjustment [63], transmission switching [27]-[28], and load shedding [64]. DC power flow based SCED is used to relieve flow violations for real-time operations of the power system [27]. Operators use tap changing transformers and phase shifting transformers to control the voltage and active power, respectively [63]. Load shedding is always used as the last option due to its adverse economic and social impacts [51].

1.2.5 Cascading Failure Analysis

Cascading failure analysis is important because of the occurrence of blackouts/brownouts all over the world at different points in time [65]. Cascading failure is a sequence of dependent failures of individual components that weakens the power grid [66]. AC power flow-based cascade failure model has been used in [67]-[68], whereas DC power flow-based cascade failure model was used in [69]-[70] for enhanced computational benefits. Dobson et al. in [71]-[72] obtained statistics of cascading line outages from utilities to understand how cascades initiate and propagate in the power system. In [73], Rezaei et al. estimated the risk of cascading failure with an algorithm called *random chemistry*. In [74], Rahnamay-Naeini et al. performed probabilistic analysis to understand the dynamics of cascading failures. In [75], Hines et al. proposed an *influence graph* model to capture patterns of cascading failures in power systems and validated the model using historical data. Instead of relying on prior historical data, which may or may not be relevant for the present

scenario, the research presented in this dissertation will exploit knowledge of the current network conditions to identify the system's critical interconnections, the loss of which might trigger a cascade. Despite different research initiatives on cascading failure analysis, there are not many non-commercial publicly available tools for researchers to simulate cascade failure analysis [76]. A DC power flow based non-commercial tool named MATCASC was developed in [76] for cascading failure analysis. MATCASC has been used in this dissertation to validate the results obtained herein.

1.2.6 Graph Theoretic Approach for Power System Vulnerability Analysis

Graph theoretic techniques have been widely used for quick assessment of power system vulnerability [77]-[92]. With regards to vulnerability assessment, graph theoretic approaches have focused on the topology and structure of the power system. Ishizaki et al. summarized the applications of graph theory for power systems modeling, dynamics, coherency, and control [77]. In [78], Albert et al. studied the structural vulnerability of the North American power grid using a metric called the *node degree*, which refers to the number of lines connected to a bus. Use of *betweenness indices*, which refer to the number of shortest paths traversing a given element, were explored in [79], [80]. These electrical betweenness indices [80] aim to find the most important transmission links with respect to the actual power flowing in the network and are governed by Kirchhoff's laws. Arianos et al. proposed a new metric called *net-ability* (a concept of distance between two nodes) to evaluate the performance of power grids [81]. Crucitti et al. used a metric called *global efficiency* of a power network to identify the critical components of the network [82]. The metric global efficiency is derived from the shortest path lengths between any two nodes

in the network [83]. On the Italian power grid, a purely topological analysis was performed by Crucitti et al. in [84]. The concept of *graph resistance* was exploited in [85] for detecting power system vulnerabilities.

Modified centrality indices were used in [86] and [87] to assess the risk of black-outs/brownouts and systemic vulnerabilities, respectively. Different statistical measures such as the *betweenness indices*, *node-degree*, and *geodesic distance* have been used as possible alternatives to power flow techniques to quantify power system vulnerability during *N*-1 contingencies and cascading failures [88], [89]. In [90], Zhu et al. proposed a metric called *risk graph* to better capture the cascade failure vulnerability of the power system. More recently, Beyza et al. investigated the structural vulnerability of the power system when successive *N*-1 contingencies progressively alter the network structure [91]. Many of the methods discussed above represent the *global vulnerability of the system* with the help of *a single index*. However, simply quantifying the global vulnerability *does not provide meaningful physical information* to a system operator because *it obscures the physical interpretation of the vulnerability* [40].

In [40], Werho et al. used a graph theory-based *network flow algorithm* to identify the cut-set of minimum size between a source-sink pair. A cut-set denotes the minimum set of branches which when removed separates the network into two disjoint islands; the size of the cut-set refers to the number of branches present in it. If the number of branches contained in the minimum sized cut-set progressively decreases, it indicates a structural weakness between the selected source-sink pair. In [92], Beiranvand et al. presented a novel topological sorting algorithm to screen out *coherent cut-sets*. Coherent cut-sets denote the set of branches that partition the network, such that the power flows in the same

direction through all the branches. However, coherent cut-sets may not be the only bottlenecks in a power system, as there may be a cut-set in which the power flows are not unidirectional, but a single outage limits the power transfer through it.

1.3 Research Scope

Building on the prior work on cut-sets in power systems [40], [92], this dissertation is aimed towards finding if a contingency will create a saturated (or overloaded) cut-set in the power network independent of the directions in which power flows through different branches of the cut-set. The complexity of the problem lies in the fact that a power system asset can be associated with innumerable cut-sets. Therefore, the research question being explored here is: *how to analyze the power transfer capability across all cut-sets associated with a transmission asset (line or transformer), and quickly screen out the cut-set that will become saturated by the largest margin as a consequence of the loss of the transmission asset?* Intelligent graph theoretic algorithms based on network science is developed in this dissertation to precisely answer this research question. Followed by the identification of saturated cut-sets due to a potential contingency, convex optimization techniques would be used to make the power system secure against saturated cut-sets. Sub-section 1.3.1 introduces the graph theoretic terminologies in the context of the power system that will be used extensively in this research. Sub-section 1.3.2 introduces saturated cut-sets with the help of an example. Finally, sub-section 1.3.3 presents the working principle for identifying saturated cut-sets due to a potential contingency.

1.3.1 Graph Theoretic Terminologies Used in Power System

The power system is represented by a graph $\mathcal{G}(V, E)$, with the buses contained in set V, and all branches (transmission lines and transformers) contained in set E [93]. The sets G and L contain all the generator (source) buses and load (sink) buses, respectively. Every transmission asset (line or transformer) is associated with a maximum power transfer capability referred to as the *asset rating*. Hence, every branch $e_l \in E$ is associated with a weight f_l^{max}, where f_l^{max} denotes the asset rating of branch e_l. For example, in Fig. 1.1 branch e_5 joining buses 4 and 5 has a flow of 15 MW from bus 4 towards bus 5 (i.e., $f_5 = 15$), and the corresponding branch rating is 250 MW (i.e., $f_5^{max} = 250$).

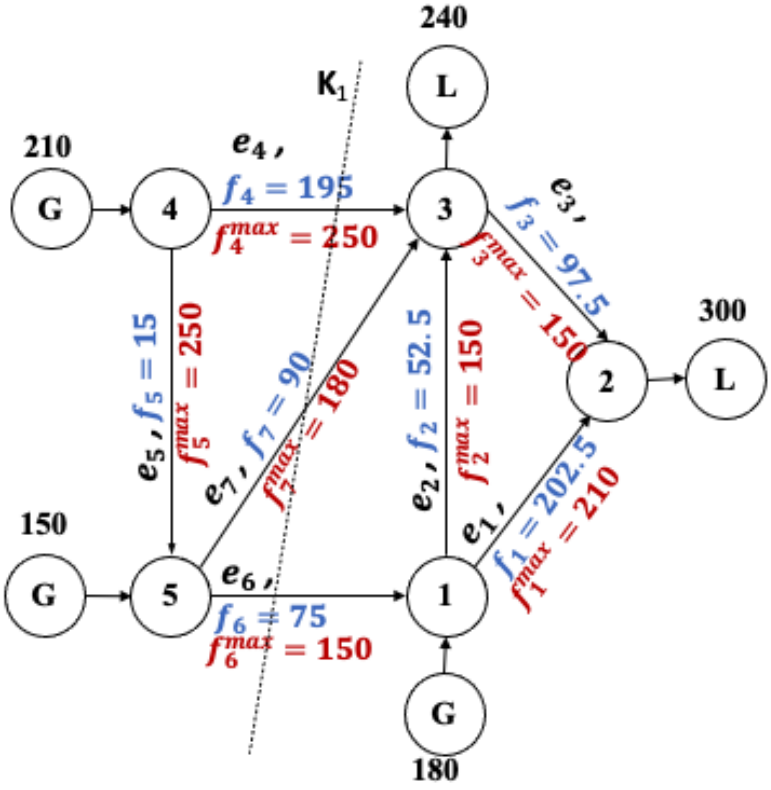

Fig. 1.1 Effect of a Contingency on a Cut-set of the Power Network

1.3.2 Introduction to Saturated Cut-sets

A cut-set is defined as the set containing minimum number of branches which when removed splits the network into two disjoint islands [93]-[94]. Any cut-set which transfers more power from one area to another than is permissible by the maximum power transfer capability of the cut-set is called a *saturated* cut-set. Let branches e_1, e_2,…, e_k belong to cut-set K. If the flows through the different branches of cut-set K are f_1, f_2,… f_k, and the ratings of those branches are f_1^{max}, f_2^{max},… f_k^{max}, cut-set K is called a saturated cut-set if the following equation holds true:

$$\sum_{l=1}^{k} f_l > \sum_{l=1}^{k} f_l^{max}, \forall e_l \in K \tag{1.9}$$

where $\sum_{l=1}^{k} f_l = P_K$ is the actual power flow occurring through cut-set K and $\sum_{l=1}^{k} f_l^{max} = R_K$ is the maximum power that can be transferred across cut-set K. The objective of this research is to find if a contingency will create a saturated cut-set in the power system. If the outage of any branch $e_l \in K$ exhausts the power transfer capability of cut-set K, then the loss of branch e_l is said to saturate cut-set K. The *transfer margin* on cut-set K for the outage of branch e_l is defined to be $R_K - P_K$. It must be noted that *for a saturated cut-set the transfer margin is negative*.

The concept of saturated cut-sets is explained with the help of Fig. 1.1. The cut-set K_1 in Fig. 1.1 contains branches e_4, e_6, and e_7; i.e., $K_1 = \{e_4, e_6, e_7\}$. Total power transferred across this cut-set is $P_{K_1} = f_4 + f_6 + f_7 = 360$ MW. The total power transfer capacity across this cut-set is $R_{K_1} = f_4^{max} + f_6^{max} + f_7^{max} = 580$ MW. It is easy to observe that the cut-set K_1 is unsaturated as $P_{K_1} < R_{K_1}$. However, the loss of branch 3-4 would

saturate cut-set K_1. This is because with the outage of branch 3-4, the power that must be transferred from Area 1 to Area 2 (to satisfy the total load with total generation) is still 360 MW (i.e., $P_{K_1} = 360$ MW), but the total power transfer capability of cut-set K_1 reduces to 330 MW (as now $R_{K_1} = f_6^{max} + f_7^{max}$). Consequently, outage of branch 3-4 saturates cut-set K_1 by 30 MW ($R_{K^1} - P_{K^1} = 330 - 360 = -30$ MW); in other words, *the transfer margin* is -30 MW.

It must be noted here that a single branch, e.g., 3-4 in Fig. 1.1, can be associated with multiple cut-sets, such as, K_2={3-4,4-5},K_3={3-4,3-5,1-3,1-2}, and K_4={3-4,3-5,1-3,2-3}. This implies that to assess the impact of the loss of any asset on any cut-set of the power system (to check whether it has become saturated or not), we must examine the power transfer capability of *all cut-sets associated with that asset*. For a big system containing thousands of buses, a single asset could be associated with hundreds of cut-sets. Therefore, quantifying the impact of an outage on any cut-set of the power network is a computationally intensive task.

1.3.3 Working Principle for Detecting Saturated Cut-sets

Detection of saturated cut-sets due to a potential contingency is based on the following idea. Let a branch e_l (transmission line or transformer) connect buses v_l^F and v_l^T as shown in Fig. 1.2. Since branch e_l is a single element that joins bus v_l^F to v_l^T it is called a direct path from bus v_l^F to v_l^T. There could be many other electrical paths to transfer power from bus $v_l^F$to bus v_l^T. Any path that contains multiple branches from v_l^F towards v_l^T is an indirect path. If all the indirect paths combined do not have the capacity to re-route f_l units of power that was flowing through the direct path, it implies that the loss of branch e_l would

inevitably result in post-contingency cut-set saturation. Based on this inference, a graph theory-based network analysis tool is developed in this dissertation to quickly detect violations of the type where the set of indirect paths do not have extra capacity to carry the power that was originally flowing through the direct path.

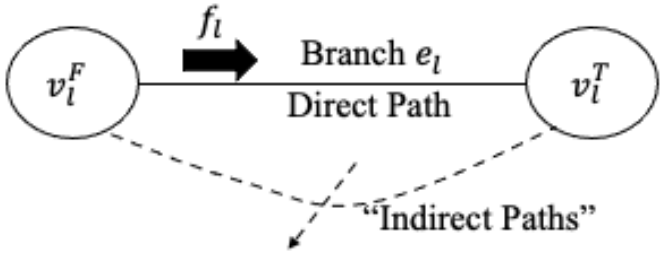

Fig. 1.2 Network Connectivity Between Two Buses

Now, let branch e_l be associated with x cut-sets of the network, of which y cut-sets ($y \leq x$) become saturated by a negative transfer margin when e_l is lost (implying that y cut-sets of the network are saturated). As the y cut-sets may be saturated by different negative transfer margins, T_l^i, $1 \leq i \leq y$, the objective here is to identify the cut-set that becomes saturated by the numerically largest negative transfer margin (i.e., $T_l = \max(|T_l^i|);\ 1 \leq i \leq y$); this cut-set is henceforth referred to as the *limiting critical cut-set*, K_{crit}. The detection of the limiting critical cut-set must be followed by a corrective action to alleviate the saturation of the identified cut-set. The detection and corrective action should be executed iteratively, such that there exists no saturated cut-sets in the network due to any contingency. Note that this dissertation will identify the limiting critical cut-sets based on the thermal ratings of the different assets and the active power flowing through them (power factor is set to unity for the studies done here). However, the proposed network analysis tool is generic enough to incorporate branch ratings obtained from other analyses as well (such as, proxy limits based on power system stability criteria).

In Section 1.2.1, we have seen that different methods of power flow analysis (AC power flow, decoupled power flow, and DC power flow) involve different modeling detail and approximations. The proposed research will investigate another level of approximation in the power flow model to study the properties of cut-set power transfers. The approximation will involve relaxation of Kirchhoff's voltage law (KVL) but will satisfy the law of conservation of energy. The flow solution will be referred to as the graph-theory based network flow solution (see Section 2.4). The relaxation of the KVL constraint will facilitate existence of multiple valid network flow solutions. However, the power transfer across any cut-set of the network will remain constant because of conservation of energy. We will observe that this relaxed graph-theory based network flow model can provide useful information on different aspects of cut-set power transfer at a significantly enhanced computational speed.

The subsequent sections of this dissertation are organized as follows. Chapter 2 presents a new graph-theory based network analysis technique which can detect post-contingency cut-set saturation. It describes the theoretical foundations of the proposed methodology by virtue of which it can achieve high computational efficiency. Chapter 3 presents the results and discussions with the help of different case-studies in the context of detecting saturated cut-sets in power networks. Chapter 4 presents a constrained optimization formulation to secure the power system against post-contingency cut-set saturation. Chapter 5 presents the results and discussion for the mitigation of saturated cut-sets with the help of different case-studies. Finally, Chapter 6 presents the concluding statements and the scope of future work.

CHAPTER 2

DETECTION OF SATURATED CUT-SETS

This Chapter presents different graph-theory based network flow algorithms to detect saturated cut-sets in power systems at enhanced computational speed. The basis of the proposed network analysis depends on intelligent graph traversal schemes on weighted graphs. The theoretical arguments introduced in this Chapter, followed by different toy examples, will demonstrate how a *relaxed* steady-state network analysis method can quickly evaluate the impact of a transmission contingency on different cut-sets in a power system.

2.1 The Flow and Latent Capacity Graphs

The flow graph, defined as $\mathcal{F}(V, E)$, contains information about power flowing through different branches of the network. Fig. 2.1(a) shows a flow graph for a sample 5 bus power system obtained from a DC power flow solution (the branch reactances for this system is available in Appendix A). The notation introduced in Section 1.3.1 is used to describe the flow graph in Fig. 2.1(a). f_l and f_l^{max} represent the flow and branch rating for the respective branches. A latent capacity graph $\mathcal{C}(V, E)$ is created from the flow graph $\mathcal{F}(V, E)$. The latent capacity graph provides information regarding the following: for any branch $e_l \in E$, *what is the extra power that could be transferred from bus* v_l^F *to* v_l^T, *and vice-versa* (v_l^F and v_l^T denote the "from bus" and "to bus" of branch e_l, respectively). The extra power transfer capability in a specific direction is called the "latent capacity" of the branch in that direction; hence, the name latent capacity graph. Each branch of the graph $\mathcal{C}(V, E)$ is associated with bidirectional weights: c_l^{FT} and c_l^{TF}, such that c_l^{FT} and c_l^{TF} denote

the "latent capacity" of branch e_l in the direction from v_l^F to v_l^T and v_l^T to v_l^F, respectively. The two weight components are given by:

$$\left.\begin{aligned} c_l^{FT} &= f_l^{max} - f_l \\ c_l^{TF} &= f_l^{max} + f_l \end{aligned}\right\} \tag{2.1}$$

Fig. 2.1(b) shows the corresponding latent capacity graph for the flow graph in Fig. 2.1(a). Branch e_1 is associated with a flow of 202.5 MW from bus 1 towards bus 2; i.e., $f_1 = 202.5$ (refer to Fig. 2.1(a)). Note that the maximum power transfer capacity of branch e_1 is 210 MW. Therefore, the extra flow that can be transferred through branch e_1 from bus 1 towards bus 2 is 7.5 MW ($= f_1^{max} - f_1 = 210 - 202.5$). On the other hand, the extra power that can be transferred from bus 2 towards bus 1 is 412.5 MW ($= f_1^{max} + f_1 = 210 + 202.5$). The same observation holds true for the latent capacities associated with other branches in the network. The latent capacity graph will be traversed exhaustively to detect saturated cut-sets in power systems.

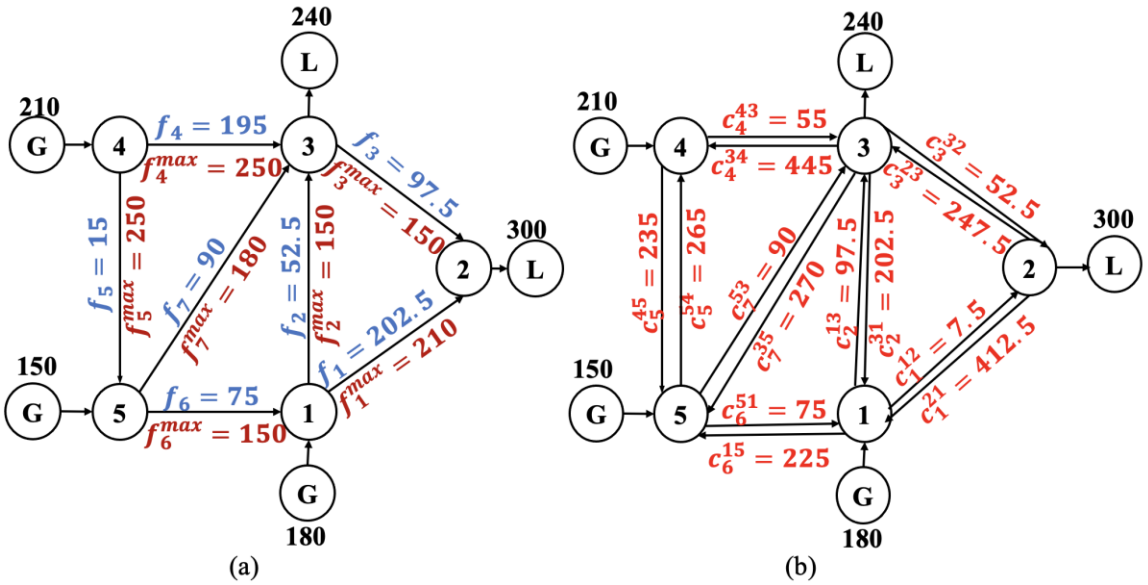

Fig. 2.1 (a) A Sample Flow Graph $\mathcal{F}(V, E)$, and (b) Latent Capacity Graph $\mathcal{C}(V, E)$ for a Sample 5-bus Power System. This Flow Solution is Obtained from a DC power flow

2.2 Saturated Branch and Saturated Paths

The concept of saturated branch and saturated paths will be extensively used in different graph-theoretic algorithms developed in this dissertation. Consider that a branch e_l connects buses v_l^F and v_l^T. If it cannot transfer additional power from v_l^F towards v_l^T, it is said to be saturated in the corresponding direction. In other words, if the latent capacity for a branch is zero in a specific direction it is said to be saturated along that direction. Now, a path contains a sequence of branches from a given source to a given sink. A saturated path will contain at least one branch that has a latent capacity of zero along the specified direction. For example, let us consider a path from bus 4 to bus 2 in Fig. 2.1(b): $\mathcal{P} = \{4 - 5 - 1 - 2\}$. In path $\mathcal{P}$, none of the branches are saturated along the direction described by the path from bus 4 towards bus 2 (see Fig. 2.1(b)). This implies that all branches in path $\mathcal{P}$ are unsaturated, and consequently path $\mathcal{P}$ is an unsaturated path.

2.3 Breadth First Search (BFS) Graph Traversal

The two most popular techniques of graph traversal are the breadth first search (BFS) algorithm [95] and the depth first search (DFS) algorithm [96]. For traversing the *shortest path* from a specific source to another sink, BFS is advantageous to DFS. This is because, when BFS is used to traverse the graph to reach the specified sink from a given source, the path traced by BFS is already the shortest path. If there had been a shorter path, BFS would have found it earlier. Moreover, the graph-theoretic algorithms that will be presented in the later sections of this dissertation will mostly depend on finding the shortest *unsaturated* path from a source bus to a sink bus in the network. Therefore, only unsaturated branches are considered during the graph traversal from the source to the sink; thereby eliminating

the possibility of selecting saturated paths. This is explained with the help of the bidirectional latent capacity graph $\mathcal{C}(\mathrm{V}, \mathrm{E})$ shown in Fig. 2.1(b). Let us assume that the *source* bus is 4 and the *sink* bus is 2. The BFS graph traversal takes place in the following steps:

Step 1: In the first step, the source bus is identified with a depth of "0".

Fig. 2.2 Step 1 of the Graph Traversal Using BFS

Step 2: Buses 3 and 5 are adjacent to bus 4, and branches 4-3 and 4-5 have non-zero latent capacities. Therefore, buses 3 and 5 are connected to bus 4 at a depth of "1" as shown in Fig. 2.3.

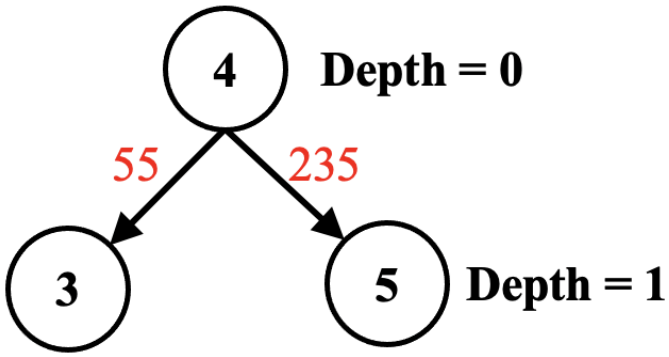

Fig. 2.3 Step 2 of the Graph Traversal Using BFS (the Latent Capacities of the Branches Along Given Direction are Shown in Red)

Step 3: Those buses which are adjacent to buses 3 and 5, but which have not been traversed yet are identified in this stage. Since, bus 3 is connected to buses 1 and 2, and bus 5 is connected to bus 1, they are added at a depth level of "2" as shown in the Fig. 2.4 below. It must also be noted that each of the branches 3-1, 3-2 and 5-1 are associated with non-zero latent capacities. Since the sink bus "2" is reached in this step, the process is not

repeated and the path from the source bus to the sink bus is back tracked to obtain the shortest unsaturated path, which is given by $\mathcal{P} = \{4 - 3 - 2\}$.

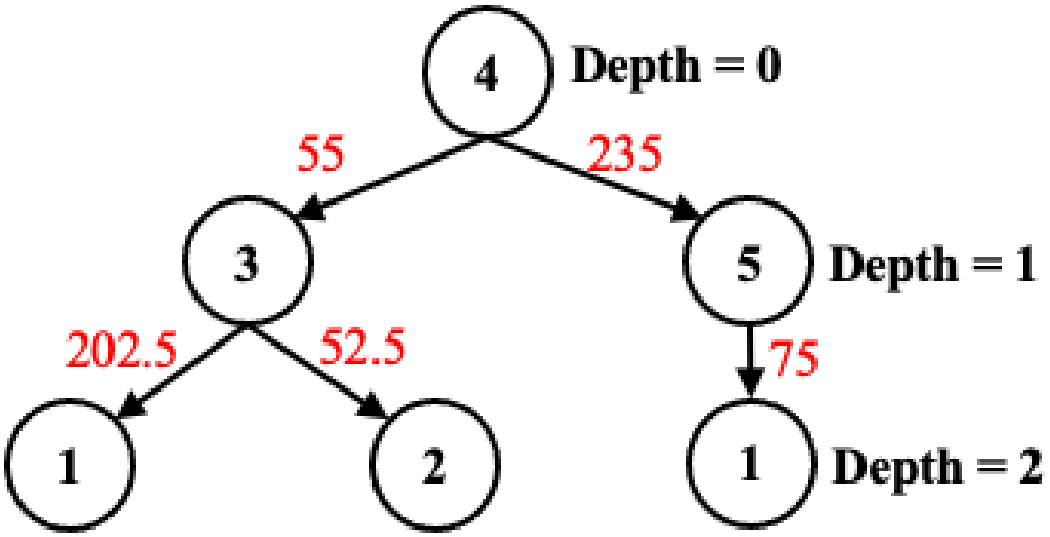

Fig. 2.4 Step 3 of the Graph Traversal Using BFS (the Latent Capacities of the Branches Along a Given Direction are Shown in Red)

In this research, we have used the BFS scheme for traversing the bi-directional latent capacity graph $\mathcal{C}(\mathrm{V}, \mathrm{E})$. The BFS function can be found in MATLAB's graph theory toolbox. If the graph $\mathcal{C}(\mathrm{V}, \mathrm{E})$ is to be traversed from a *source* bus to a *sink* bus, the algorithm starts at the source bus, and explores all the neighboring buses at the present depth prior to moving on to the buses at the next depth. Once the sink is reached the algorithm stops.

2.3.1 Time Complexity of Shortest Path Graph Traversal Schemes

Apart from BFS, other commonly used graph traversal methods for finding the shortest path between a source-sink pair are Bellman-Ford algorithm [97], and Dijkstra algorithm [98]. If $|E|$ denotes the total number of branches, and $|V|$ denotes the total number buses, the time-complexity of the Bellman-Ford algorithm is O(|E||V|) [99]. The time-complexity of Dijkstra algorithm implemented using binary heap is $O(|E| + |V|log|V|)$ [100].

Lastly, the time-complexity of the BFS algorithm is $O(|E| + |V|)$ [101], which is the best among the three shortest-path graph traversal techniques. Therefore, we have used the BFS graph traversal scheme to develop different algorithms to determine if contingencies create saturated cut-sets in power networks.

2.4 Graph-theory based Network Flow Algorithm (NFA)

The graph theory-based network flow algorithm is based on the following assumptions: (1) power injections are known, and (2) losses are negligibly small. Subject to these assumptions, the goal is to generate network flows that can help detect if a contingency saturates a cut-set. The graph theoretic network flow algorithm is based on the following principle: *utilize the available generation of the sources (generators) to satisfy the total demand of the sinks (loads), without violating the asset ratings*. The network flows are obtained using Algorithm I described below. At the start of the algorithm, branches in $\mathcal{F}(V, E)$ do not have any weight, while the bidirectional weights of the branches in $\mathcal{C}(V, E)$ are equal to the corresponding asset ratings.

The graph theory-based network flow algorithm obeys the law of conservation of energy, but it relaxes KVL as it does not use impedances *directly* while building the network flows; the impedances are accounted for indirectly through the asset ratings. The flow solution is also non-unique because depending on the order in which the sources and sinks are selected, there could be multiple valid flow solutions. However, the power transfer across any cut-set of the network is the same for all valid graph-theory based network flow solutions. This is explained as follows.

Algorithm I: Graph theory-based Network Flow Algorithm (NFA)

i. Randomly select a source bus $v_i \in G$ and a sink bus $v_j \in L$.

ii. Search $\mathcal{C}(V, E)$ to traverse the shortest unsaturated path $\mathcal{P}$ from v_i to v_j using breadth first search (BFS) [95].

iii. Use $\mathcal{C}$ to find the maximum extra flow, $C_{\mathcal{P}}$, that could be transferred from v_i to v_j through path $\mathcal{P}$.

iv. Obtain the flow $F_{\mathcal{P}}$ to be injected in $\mathcal{F}(V, E)$ along path $\mathcal{P}$ from v_i to v_j as $F_{\mathcal{P}} = \min(G_i, L_j, C_{\mathcal{P}})$; where G_i is the active power generated at source v_i and L_j is the active power demand at sink v_j.

v. Update the weights of branches in $\mathcal{F}(V, E)$ as $f_l = f_l + F_{\mathcal{P}}$, and in graph $\mathcal{C}(V, E)$ as $c_l^{FT} = c_l^{FT} - F_{\mathcal{P}}$ and $c_l^{TF} = c_l^{TF} + F_{\mathcal{P}}$ for all branches that belong to path $\mathcal{P}$.

vi. Update the available generation and unsatisfied demand at buses v_i and v_j as $G_i := G_i - F_{\mathcal{P}}$ and $L_j := L_j - F_{\mathcal{P}}$.

vii. Depending upon the values of G_i and L_j, update the source and sink buses in accordance with the following logic:

   a. if $G_i \neq 0$ & $L_j \neq 0$, the source and sink buses are not changed.

   b. if $G_i = 0$ & $L_j \neq 0$, a new source, v_i, is selected from G, keeping the sink, v_j, unchanged.

   c. if $G_i \neq 0$ & $L_j = 0$, a new sink, v_j, is selected from L, keeping the source, v_i, unchanged.

viii. Repeat Steps (ii) through (vii) until the total power generation satisfies the total power demand.

Let the network graph $\mathcal{G}(V, E)$ be split into two clusters C_1 and C_2 such that $C_1 \cup C_2 = V$ and $C_1 \cap C_2 = \emptyset$ as shown in Fig. 2.5. If $P_G^1(P_G^2)$ and $P_L^1(P_L^2)$ be the total generation and total demand in $C_1(C_2)$, then the net generation in C_1 is given by $\Delta P^1 = P_G^1 - P_L^1$, while the net generation in C_2 is given by $\Delta P^2 = P_G^2 - P_L^2$. Now, cut-set K between clusters C_1 and C_2 would include only those branches whose one end belongs to C_1 and the other end belongs to C_2; let the number of branches in cut-set K be k. Also, let $f_1^A, f_2^A, \ldots, f_k^A$ denote network flows through different branches of cut-set K for a valid graph-theory based flow solution A, and $f_1^B, f_2^B, \ldots, f_k^B$ denote the network flows through the same branches for a valid graph-theory based flow solution B. Then, by the law of conservation of energy, total power transfer across cut-set K for each of the flow solutions A and B must be equal to $\Delta P_1 = -\Delta P_2$, i.e.,

$$\sum_{l=1}^{k} f_l^A = \sum_{l=1}^{k} f_l^B = \Delta P^1 = -\Delta P^2, \qquad \forall\, e_l \in K \qquad (2.2)$$

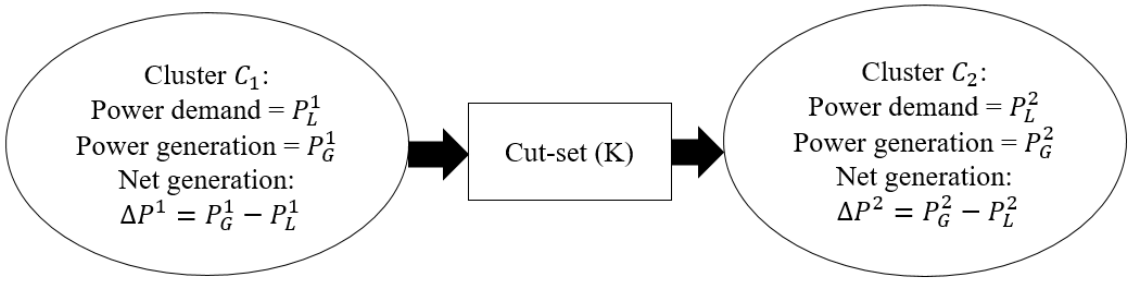

Fig. 2.5 The Original Power Network is Divided into Two Disjoint Clusters C_1 and C_2

2.4.1 Graph-theory based Network Flows on a Sample 5-bus Test System

For the sample 5-bus system depicted in Fig. 2.1 the sets G (containing source locations) and L (containing sink locations) comprise of: $G = \{1,4,5\}$ and $L = \{2,3\}$. At the start of the solution, the network flows through different branches of the system are

initialized to zero. The corresponding *flow* and *latent capacity graphs* $\mathcal{C}(\mathrm{V},\mathrm{E})$ are depicted in Fig. 2.6(a) and Fig. 2.6(b) respectively.

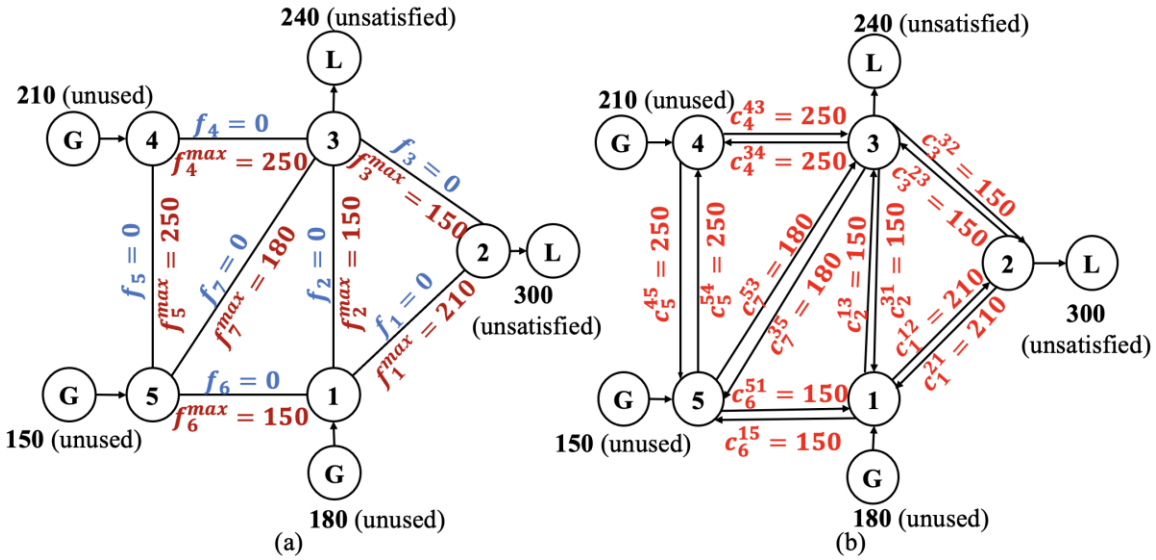

Fig. 2.6 (a) Flow Graph and (b) Latent Capacity Graph of the 5-bus Test System at the Beginning of the Network Flow Algorithm

Iteration 1:

Step i: A source is selected from set G randomly, say, $v_i = 5$. A sink is selected from set L randomly, say, $v_j = 3$. Therefore, power generation at the source and sink buses are 150 MW and 240 MW, respectively, i.e., $G_i = 150$ MW and $L_j = 240$ MW.

Step ii: The shortest unsaturated path from the source bus 5 towards sink bus 3 is given by $\mathcal{P} = \{5 - 3\}$.

Step iii: The maximum power that could be transferred through path $\mathcal{P}$ from source bus 5 towards sink bus 3 is limited by 180 MW, i.e., $C_{\mathcal{P}} = 180$ (see Fig. 2.6(b)).

Step iv: Now, the flow $F_{\mathcal{P}}$ that will be injected along path $\mathcal{P}$ of the flow graph $\mathcal{F}(V,E)$ from source bus 5 towards sink bus 3 is given as follows.

$$F_{\mathcal{P}} = Min\left(G_i, L_j, C_{\mathcal{P}}\right) = Min(150,240,180) = 150 \qquad (2.1)$$

Step v: The flow and latent capacity graphs (see Fig. 2.7) are updated for an injection of 150 MW of flow along path $\mathcal{P}$.

Step vi: Accordingly, the available generation and the unsatisfied power demand at the source bus 5 and sink bus 3 are given as follows:

$$\left.\begin{aligned} G_i &= G_i - F_{\mathcal{P}} = 150 - 150 = 0 \\ L_j &= L_j - F_{\mathcal{P}} = 240 - 150 = 90 \end{aligned}\right\} \tag{2.2}$$

After *Step vi*, the flow and latent capacity graphs are shown in Fig. 2.7.

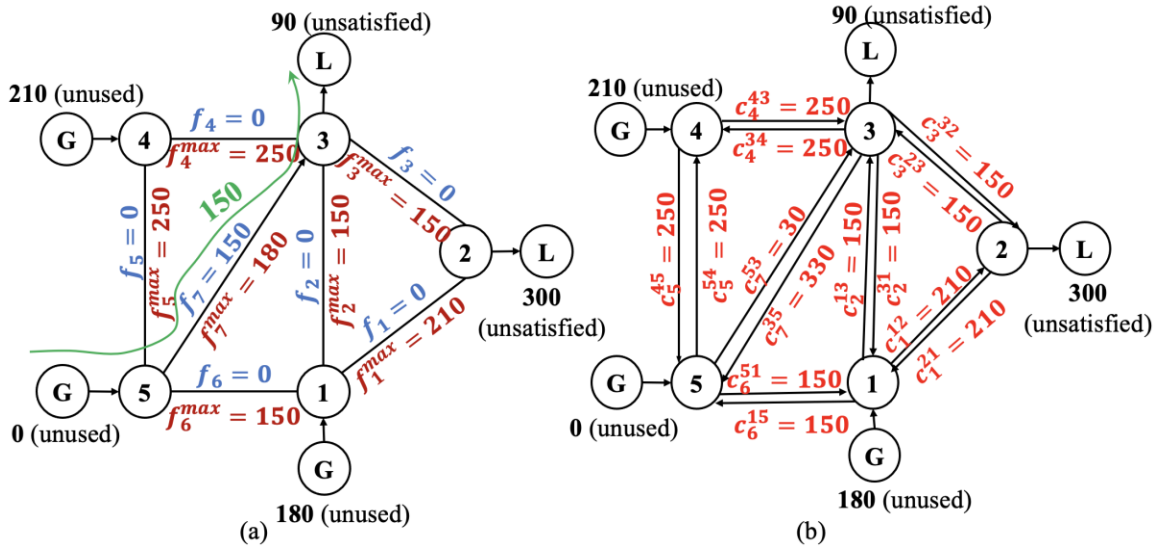

Fig. 2.7 Iteration 1-(a) Flow Graph, and (b) Latent Capacity Graph of the 5-bus Test System

Step vii: Since $G_i = 0$ and $L_j \neq 0$, a new source v_i is selected keeping the sink v_j un-changed. Let the new source be bus 4, i.e., $v_i = 4$ and $v_j = 3$. The new values of G_i and L_j are 210 MW and 90 MW respectively.

Iteration 2:

Step ii: The shortest unsaturated path, which is selected from the source bus 4 to the sink bus 3 is given by path $\mathcal{P} = \{4 - 3\}$.

Step iii: The maximum power that could be transferred from the source to the sink through path $\mathcal{P}$ is 250 MW, i.e., $C_{\mathcal{P}} = 250$ (see Fig. 2.7(b)).

Step iv: Now, the flow $F_{\mathcal{P}}$ that will be injected along path $\mathcal{P}$ is given by (2.3).

$$F_{\mathcal{P}} = Min\left(G_i, L_j, C_{\mathcal{P}}\right) = Min(210,90,250) = 90 \tag{2.3}$$

Step v: The flow and latent capacity graphs are updated for an injection of 90 MW of flow along path $\mathcal{P}$ (see Fig. 2.8).

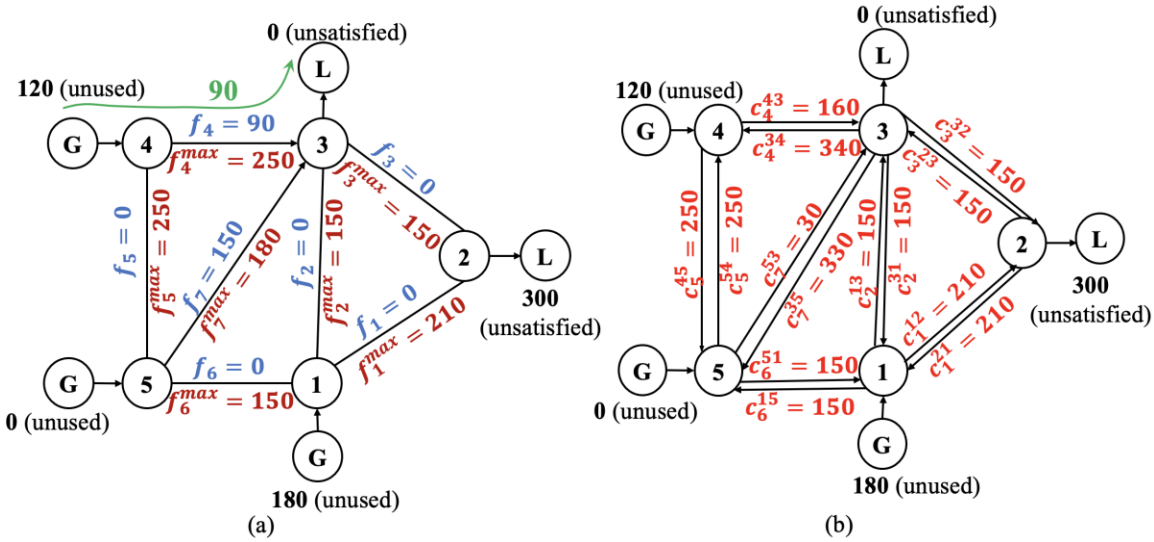

Fig. 2.8 Iteration 2- (a) Flow Graph and (b) Latent Capacity Graph of the 5-bus Test System

Step vi: The flow $F_{\mathcal{P}}$ injected through path $\mathcal{P}$ of the flow graph, and the bidirectional weights of the latent capacity graphs are now updated. Accordingly, the available generation and the unsatisfied power demand at the source and sink buses are as follows:

$$\left.\begin{aligned} G_i = G_i - F_{\mathcal{P}} = 210 - 90 = 120 \\ L_j = L_j - F_{\mathcal{P}} = 90 - 90 = 0 \end{aligned}\right\} \tag{2.4}$$

Step vii: Since $G_i \neq 0$ and $L_j = 0$, a new sink is selected from the set L keeping the source unchanged. Let the new sink be bus 2, i.e., $v_i = 4$ and $v_j = 2$. The new values for G_i and L_j are 120 and 300 MW, respectively.

Iteration 3:

Step ii: The shortest unsaturated path from the source bus 4 to the sink bus 3 is given by path $\mathcal{P} = \{4 - 3 - 2\}$ (see Fig. 2.8(b)).

Step iii: The maximum flow that could be transferred from the source to the sink through path $\mathcal{P}$ is 150 MW, i.e., $C_{\mathcal{P}} = 150$, because branch $3 - 2$ has a latent capacity of 150 MW in the direction from bus 3 towards bus 2 (see Fig. 2.8(b)).

Step iv: Now, the flow $F_{\mathcal{P}}$ that will be injected in the flow graph along path $\mathcal{P}$ is as follows:

$$F_{\mathcal{P}} = Min\left(G_i, L_j, C_{\mathcal{P}}\right) = Min(120,300,150) = 120 \tag{2.5}$$

Step v: The flow and latent capacity graphs are updated for an injection of 120 MW of flow along path $\mathcal{P}$ (see Fig. 2.9).

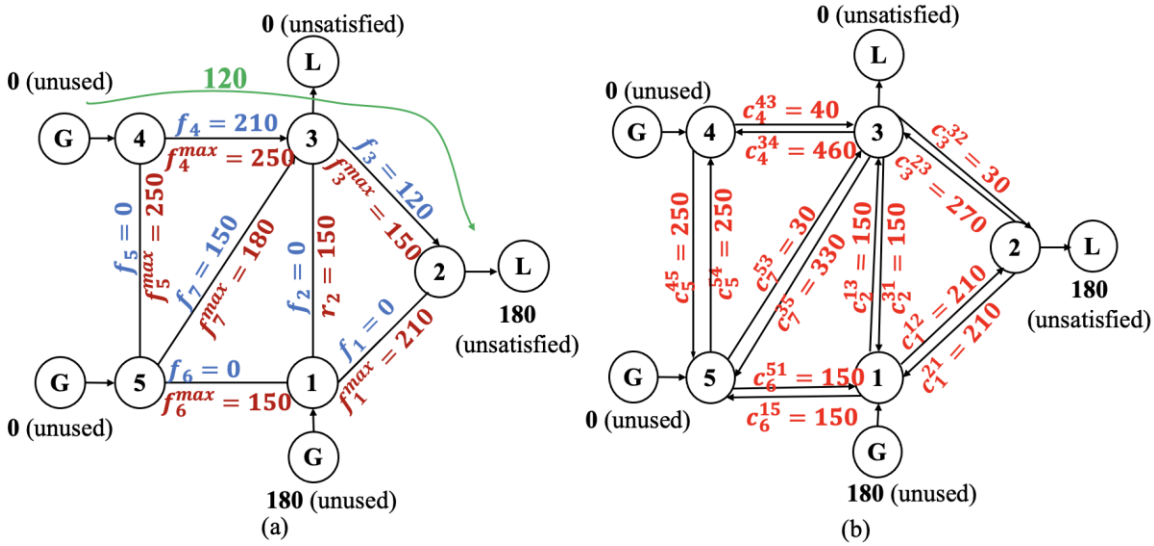

Fig. 2.9 Iteration 3-(a) Flow Graph and (b) Latent Capacity Graph of the 5-bus Test System

Step vi: Accordingly, the available generation and the unsatisfied power demand at the source and sink buses are given as follows:

$$\left.\begin{aligned} G_i &= G_i - F_{\mathcal{P}} = 120 - 120 = 0 \\ L_j &= L_j - F_{\mathcal{P}} = 300 - 120 = 180 \end{aligned}\right\} \tag{2.6}$$

Step vii: Since $G_i = 0$ and $L_j \neq 0$, the source is updated while keeping the sink unchanged. Therefore, the source and sink buses for the next iteration are buses 1 and 2, respectively, i.e., $v_i = 1$ and $v_j = 2$.

Iteration 4:

Step ii: The shortest unsaturated path from the source bus 1 to the sink bus 2 in the latent capacity graph $\mathcal{C}(\mathrm{V}, \mathrm{E})$ is given by path $\mathcal{P} = \{1 - 2\}$ (see Fig. 2.9(b)).

Step iii: The maximum power that could be transferred from source bus 1 towards sink bus 2 is 210 MW, i.e., $C_{\mathcal{P}} = 210$, because the branch $1 - 2$ has a latent capacity of 210 MW, in the direction from bus 1 towards bus 2 (see Fig. 2.9(b)).

Step iv: Now, the flow $F_{\mathcal{P}}$ that will be injected in the flow graph along path $\mathcal{P}$ is given as follows:

$$F_{\mathcal{P}} = Min\left(G_i, L_j, C_{\mathcal{P}}\right) = Min(180,180,210) = 180 \tag{2.7}$$

Step v: The flow and latent capacity graphs are updated for an injection of 180 MW of flow along path $\mathcal{P} = \{1 - 2\}$ (see Fig. 2.10).

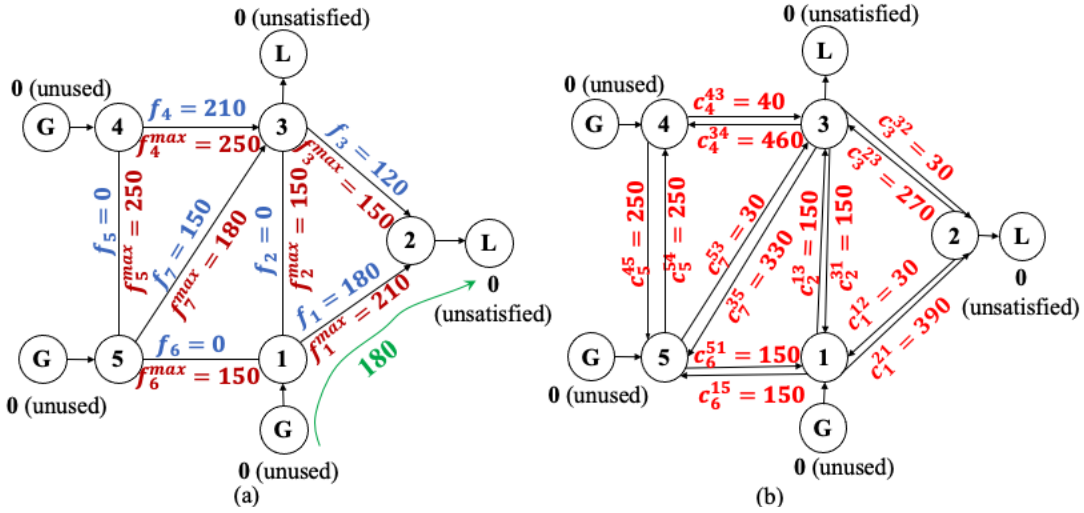

Fig. 2.10 Iteration 4-(a) Flow Graph and (b) Latent Capacity Graph of the 5-bus Test System

Step vi: Accordingly, the available generation and the unsatisfied demand at the source and sink buses are given as follows:

$$\left.\begin{aligned} G_i &= G_i - F_{\mathcal{P}} = 180 - 180 = 0 \\ L_j &= L_j - F_{\mathcal{P}} = 180 - 180 = 0 \end{aligned}\right\} \tag{2.8}$$

Step vii: Since $G_i = 0$ and $D_j = 0$, and the total load is satisfied by the total available generation, the final flow and the latent capacity graphs are depicted in Fig. 2.11.

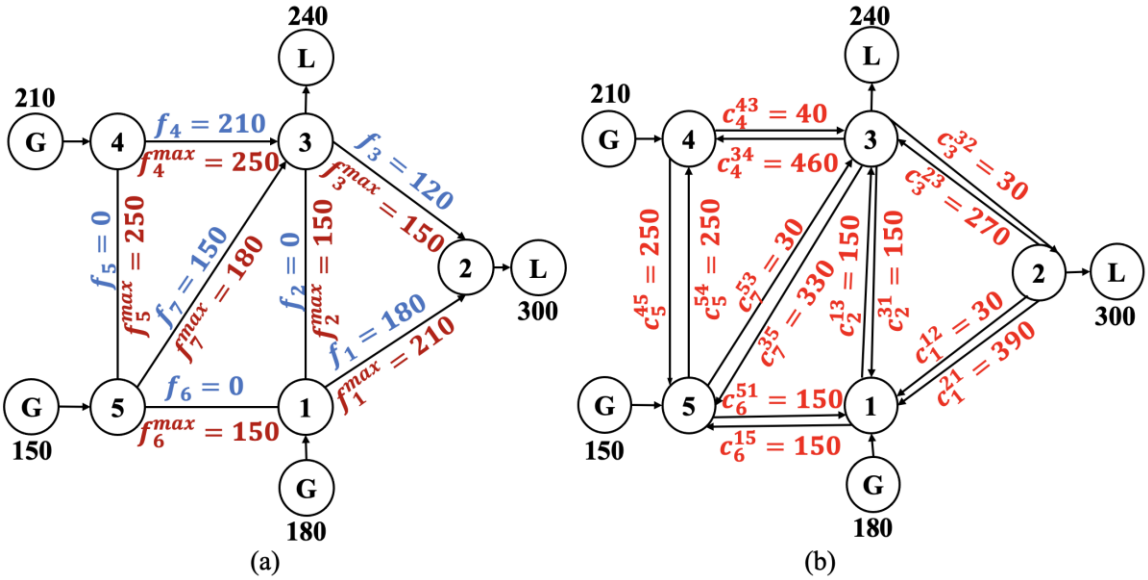

Fig. 2.11 Final Graphs-(a) Flow Graph and (b) Latent Capacity Graph of the 5-bus Test System

2.4.2 Existence of Multiple Valid Network Flow Solutions

It must be pointed out here that using the graph-theory based network flow algorithm described above, there will be multiple valid graph-theoretic network flow solutions. The base-case network flow depends upon the generators and loads that were selected in the different iterations of the network flow algorithm. For example, another valid graph-theoretic flow solution for the same system is given in Fig. 2.12(a). Fig. 2.12(b) shows the

corresponding latent capacity graph. The flow solution of Fig. 2.12 is obtained from the following iterations:

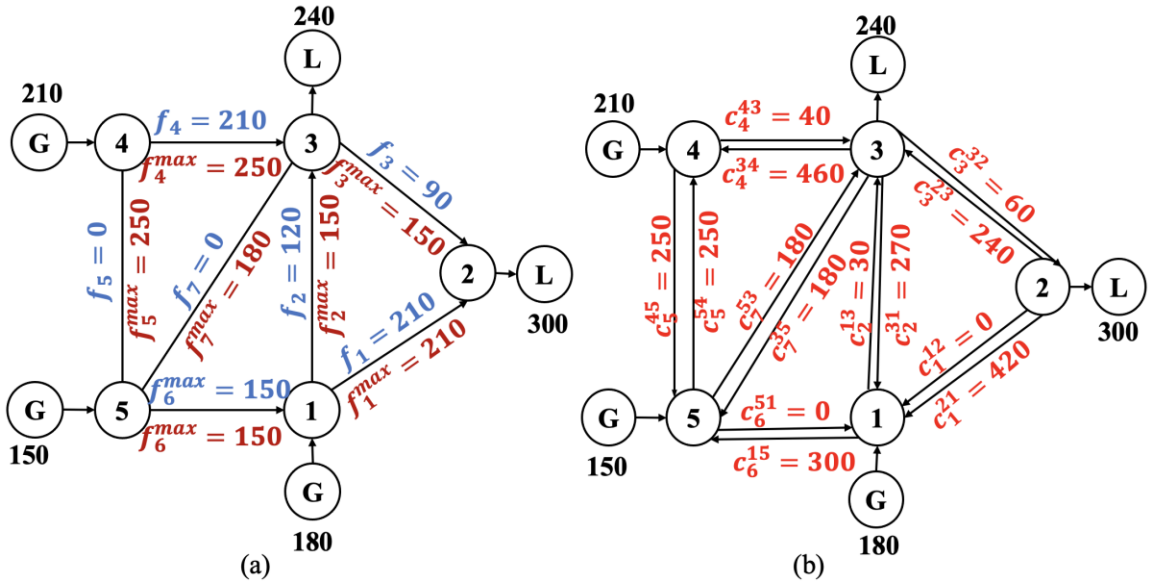

Fig. 2.12 Final Graphs-(a) Flow Graph and (b) Latent Capacity Graph of the 5-bus Test System

Iteration 1- 150 MW of power generation at source bus 5 is used to satisfy 150 MW of power demand at the sink bus 2 via path 5-1-2.

Iteration 2- 60 MW of power generation at the source bus 1 is used to satisfy 60 MW of power demand at the sink bus 2 via path 1-2.

Iteration 3- 90 MW of power generation at the source bus 1 is used to satisfy remaining 90 MW of power demand at the sink bus 2 via path 1-3-2.

Iteration 4- 30 MW of remaining power generation from source bus 1 is used to satisfy 30 MW of power demand at the sink bus 3 via path 1-3.

Iteration 5- 210 MW of power generation from source bus 4 is used to satisfy the remaining 210 MW of power demand at the sink bus 3 via path 4-3.

Fig. 2.13 compares the three valid flow graphs for the same system. Fig. 2.13(a) shows a flow solution obtained from DC power flow solution. The corresponding branch

reactance that were used in the DC power flow solution is available in Appendix A. Fig. 2.13(b) shows a valid graph theory-based network flow solution. Fig. 2.13(c) shows another valid graph-theory based network flow solution. We can see that the individual branch flows are different for different flow solutions. However, the total power transferred across any cut-set in the network is constant. For example, the total power transferred across cut-set *K* is 360 MW, for each flow graphs as enumerated in Table 2.1.

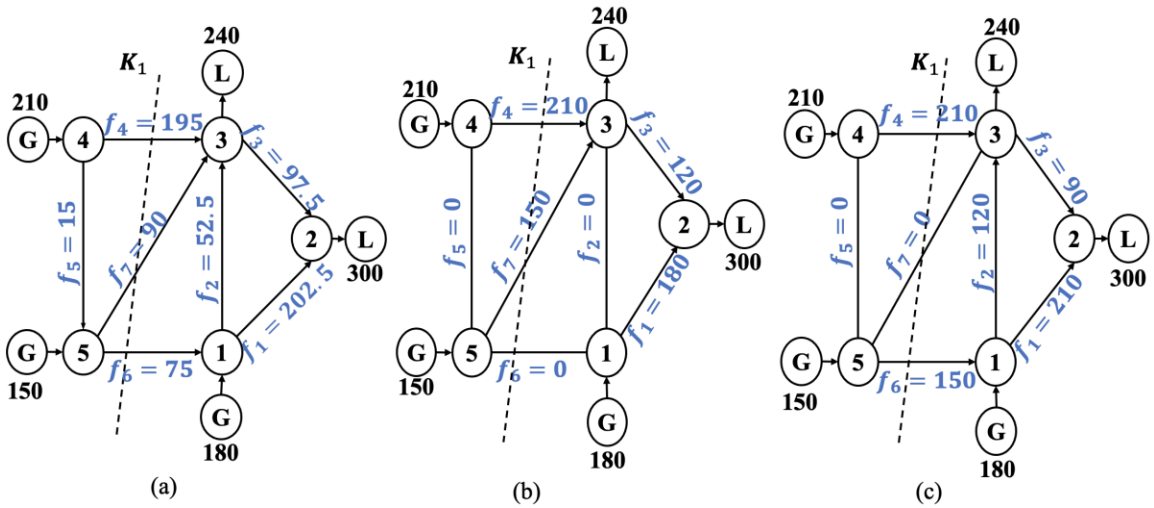

Fig. 2.13 (a) *Case A*: A Flow Graph Obtained From a DC Power Flow Solution, (b) *Case B*: A Flow Graph Obtained From a Valid Graph-Theory Based Network Flow Solution, (c) *Case C*: A Flow Graph Obtained From Another Valid Graph-Theory Based Network Flow Solution for the 5-bus Test System

Table 2.1 Power Transfer Across a Cut-set for Three Different Network Flow Solutions of a 5-bus Power System

Branches in K_1	*Case A*: Flow (MW)	*Case B*: Flow (MW)	*Case C*: Flow (MW)
4-3	195	210	210
5-3	90	150	0
5-1	75	0	150
Total Power Transfer across cut-set K_1	360	360	360

Another example that demonstrates how any graph-theory based network flow algorithm generates constant flows across a cut-set independent of different branch flows is presented here. Fig. 2.14 presents a DC power flow solution for a 10-bus test system.

Branch reactance information of this 10-bus power system is presented in Appendix B. Fig. 2.15 and Fig. 2.16 present two valid graph-theory based network flow solution for the system. The ratings for each branch for this 10-bus power system is 300 MVA. Let us consider cut-set K_1 ={4-1,9-2,9-3}. Table 2.2 enumerates that the total power transferred across this cut-set is 380.86 MW independent of the fact that the individual branch flows are different for the respective flow solutions.

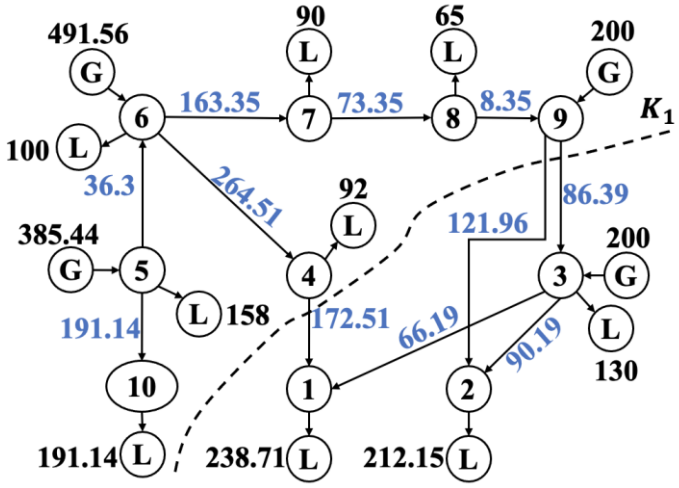

Fig. 2.14 *Case 1*- A Flow Graph Obtained from a DC Power Flow Solution (the Numbers in Blue Font on Each Branch Represent Flows) for a 10-bus Test System. The Rating for Every Branch is 300 MVA.

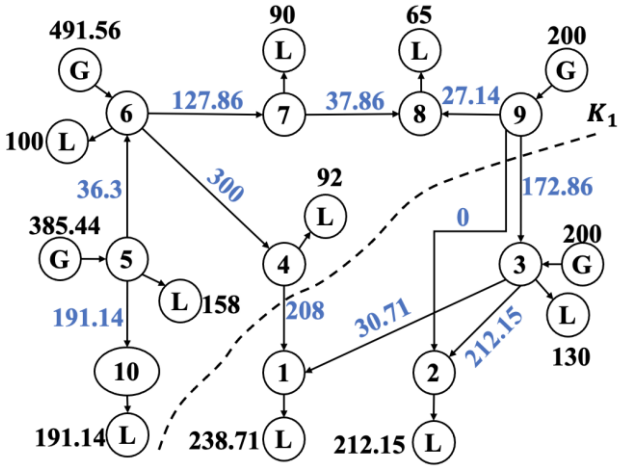

Fig. 2.15 *Case 2*- A Flow Graph Obtained from Another Graph-theory Based Network Flow Solution (the Numbers in Blue Font on Each Branch Represent Flows) for a 10-bus Test System. The Rating for Every Branch is 300 MVA.

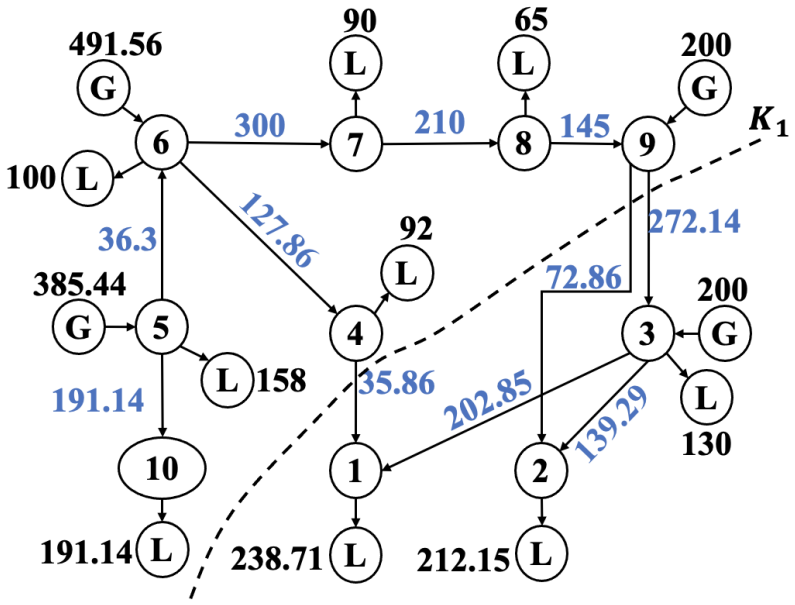

Fig. 2.16 *Case 3*- Another Flow Graph Obtained from Another Graph-Theory Based Network Flow Solution (the Numbers in Blue Font on Each Branch Represent Flows) for a 10-bus Test System. The Rating for Every Branch is 300 MVA.

Table 2.2 Power Transfer Across a Cut-set for Three Different Network Flow Solutions of a 10-bus Power System

Branches in cut-set K_1	Case 1: Power flow (MW)	Case 2: Power flow (MW)	Case 3: Power flow (MW)
4-1	172.51	208	35.86
9-2	121.96	0	72.86
9-3	86.39	172.86	272.14
Total power flow through the cut-set	380.86	380.86	380.86

2.5 Feasibility Test (FT) Algorithm

Once the graph-theory based network flows are obtained, the next step is to find out the "special assets" from the graphs $\mathcal{F}(V, E)$ and $\mathcal{C}(V, E)$. In the proposed research, the "special assets" are defined as follows: *a transmission line or a transformer is classified as special if there is no way to reroute the power flowing through it via the set of indirect paths*. The detection of the special assets is performed by the Feasibility Test (FT) algorithm, which checks for the feasibility of re-routing the power flowing through any

transmission asset via the set of indirect paths. For branch e_l, let f_l units of power flow from v_l^F to v_l^T. The steps of the FT algorithm for branch e_l are given in Algorithm II.

Algorithm II: Graph theory-based Feasibility Test (FT)

---

i. Define $\mathcal{C}'(V,E) = \mathcal{C}(V,E)$. Remove branch e_l from $\mathcal{C}'$. Initialize a variable TC_l to zero (i.e., $TC_l \coloneqq 0$).

ii. Search $\mathcal{C}'$ to obtain the shortest unsaturated path $\mathcal{P}$ from v_l^F to v_l^T using breadth first search (BFS) [95]; path $\mathcal{P}$ is considered unsaturated if it has capacity to reroute additional flow.

iii. Find the maximum extra flow, $C_{\mathcal{P}}$, that can be rerouted through path $\mathcal{P}$ from v_l^F to v_l^T.

iv. Update TC_l as $TC_l \coloneqq TC_l + C_{\mathcal{P}}$, and the weights of $\mathcal{C}'$ as follows: $c_l^{FT} = c_l^{FT} - C_{\mathcal{P}}$ and $c_l^{TF} = c_l^{TF} + C_{\mathcal{P}}$. Note that this step saturates path $\mathcal{P}$ in $\mathcal{C}'$.

v. Repeat Steps (ii) through (iv) until there exists no unsaturated path in $\mathcal{C}'$ from v_l^F to v_l^T.

vi. Due to outage of branch e_l, compute the transfer margin, T_l, as: $T_l = TC_l - f_l$. If T_l for branch e_l is negative, e_l is a special asset.

vii. To identify K_{crit}, traverse the saturated graph $\mathcal{C}'$ from v_l^F towards v_l^T. All the buses that can be reached from v_l^F without traversing a saturated branch are grouped into cluster C_1. Similarly, the buses that cannot be reached from v_l^F without traversing a saturated branch are grouped into cluster C_2. Cut-set K_{crit} contains the branches whose one end is in C_1 and the other end is in C_2.

2.5.1 Illustration of the FT Algorithm

Let us consider a valid graph-theory based network flow solution of the 5-bus test system (shown in Fig. 2.17). The FT is an iterative graph search algorithm applied on the latent capacity graph. The original latent capacity graph $C(V,E)$ is assigned to $C'(V,E)$, because the FT will make incremental changes to the latent capacity graph to evaluate the impact of the contingency. Consider that the outage of branch 3-4 is to be analyzed by the FT. From the flow graph of Fig. 2.17(a), it can be realized that the flow through branch 3-4 is 210 MW, i.e., f_l =210. The working of the FT algorithm through the different iterations is explained below.

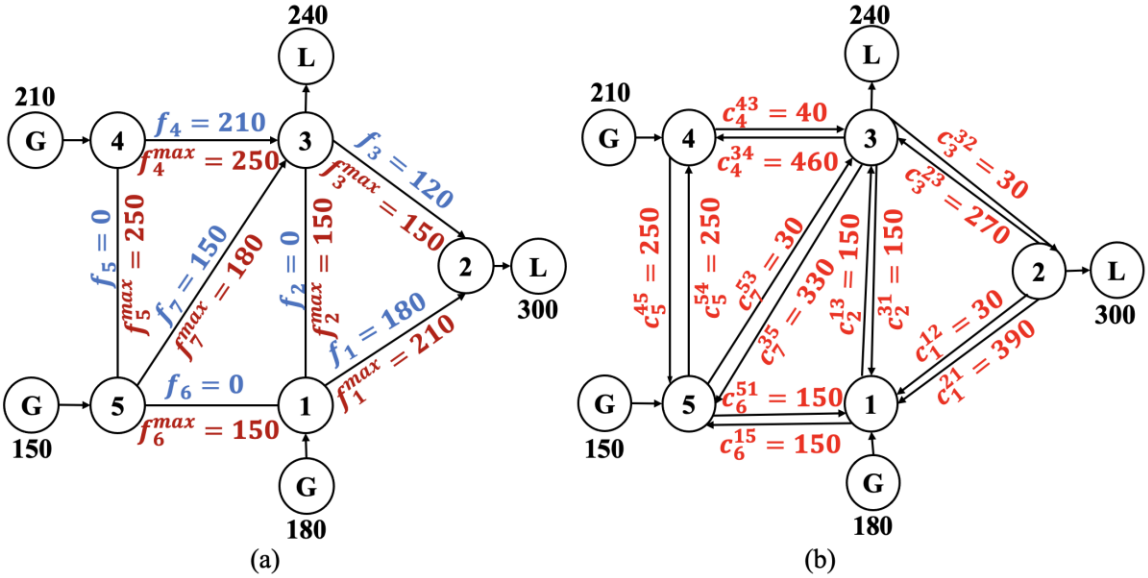

Fig. 2.17 (a) Flow Graph, and (b) Latent Capacity Graph for the 5-bus Test System Obtained from the Graph Theory-based Network Flow Algorithm

Iteration 1:

Step i: The branch 3-4 has been removed from the latent capacity graph (see Fig. 2.18). The variable TC_l is initialized to zero, i.e., $TC_l = 0$.

Step ii: The shortest unsaturated path in the latent capacity graph from bus 4 to bus 3 is given by path $\mathcal{P}$ ={4-5-3}.

Step iii: The maximum power that could be transferred through path, $\mathcal{P}$ is 30 MW (i.e., $C_{\mathcal{P}} = 30$), which is limited by branch 5-3 (see Fig. 2.18).

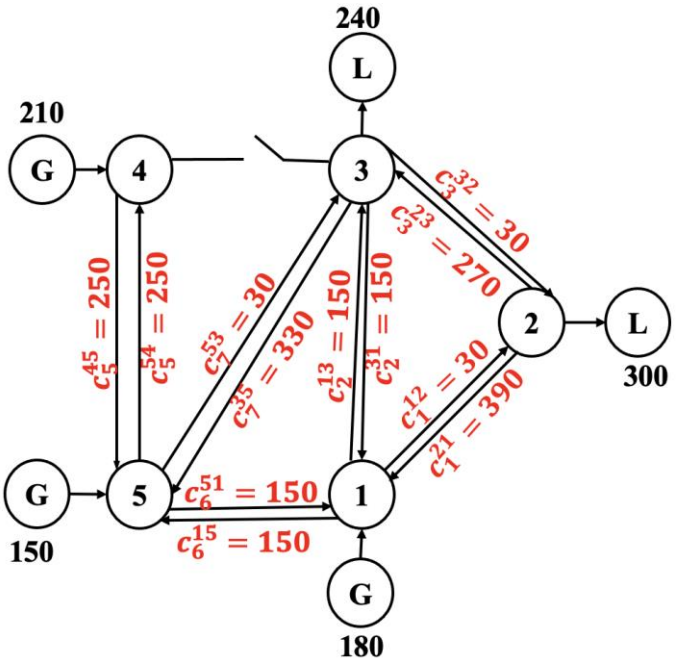

Fig. 2.18 The Branch Which is to be Evaluated for an Outage by the FT is Removed from the Latent Capacity Graph $\mathcal{C}'(V, E)$ as the First Step

Step iv: Next, TC_l is updated as follows:

$$TC_l = TC_l + C_{\mathcal{P}} = 0 + 30 = 30 \tag{2.9}$$

Moreover, the weights of the latent capacity graph $\mathcal{C}'(V, E)$ are updated for branches along path $\mathcal{P}$ to account for an injection of 30 MW of flow along the path $\mathcal{P}$. In this example, branches 4-5 and 5-3 belong to path $\mathcal{P}$. The original latent capacities (in Fig. 2.18) are $c_5^{45} = 250$, $c_5^{54} = 250$, $c_7^{53} = 30$, $c_7^{35} = 330$. As per Algorithm II, these weights are up-dated as follows: $c_5^{45} = c_5^{45} - C_{\mathcal{P}} = 250 - 30 = 220$, $c_5^{54} = c_5^{54} + C_{\mathcal{P}} = 250 + 30 =$

280, $c_5^{53} = c_5^{53} - C_{\mathcal{P}} = 30 - 30 = 0$, and $c_5^{35} = c_5^{35} + C_{\mathcal{P}} = 330 + 30 = 360$. These updated weights are shown in the updated latent capacity graph in Fig. 2.19.

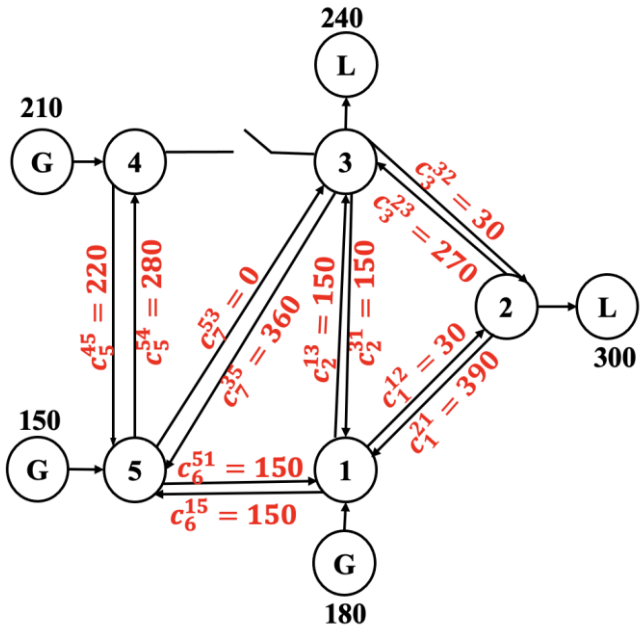

Fig. 2.19 An Updated Latent Capacity Graph $\mathcal{C}'(V, E)$ After Adding a Flow of 30 MW Along Path $\mathcal{P} = \{4 - 5 - 3\}$

Iteration 2:

Step ii: The next shortest unsaturated path in the latent capacity graph $\mathcal{C}'(\mathrm{V}, \mathrm{E})$ from bus 4 to bus 3 is given by path $\mathcal{P} = \{4 - 5 - 1 - 3\}$.

Step iii: The maximum power that could be re-routed through path $\mathcal{P}$ is 150 MW (i.e., $C_{\mathcal{P}} =$ 150 MW), which is limited by branches 5-1 and 1-3 (see Fig. 2.19).

Step iv: TC_l is updated as follows:

$$TC_l = TC_l + C_{\mathcal{P}} = 30 + 150 = 180 \tag{2.10}$$

The weights of the latent capacity graph $\mathcal{C}'(V, E)$ is updated for an injection of $C_{\mathcal{P}}$ units of flow along path $\mathcal{P}$ (see Fig. 2.20).

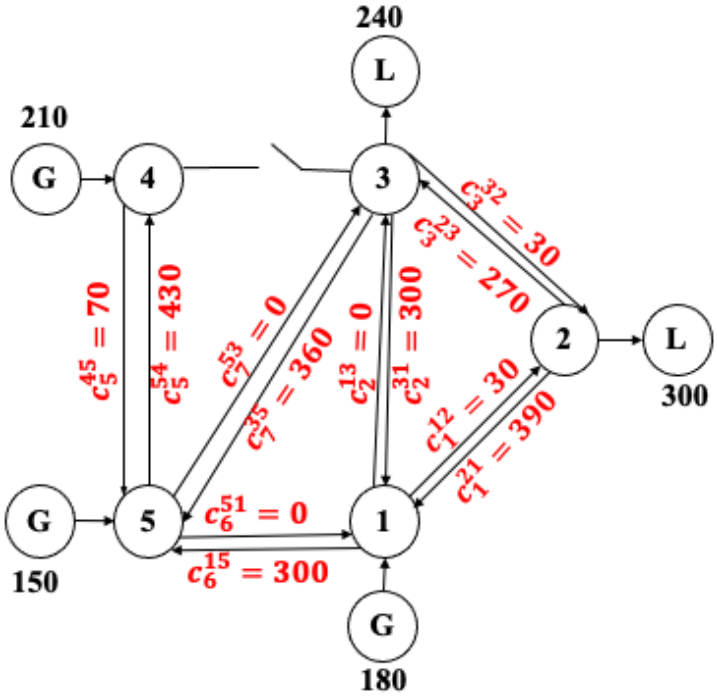

Fig. 2.20 The Updated Latent Capacity Graph $\mathcal{C}'(V, E)$ After Adding 150 MW of Flow Along Path $\mathcal{P} = \{4 - 5 - 1 - 3\}$

Step v: Since there exist no other unsaturated indirect paths from bus 4 to bus 3, the iterations are terminated.

Step vi: Now, the transfer margin, T_l for the potential outage of branch 3-4 is given by:

$$T_l = TC_l - f_l = 180 - 210 = -30 \tag{2.11}$$

Step vii: The last step of the FT algorithm identifies the limiting critical cut-set. The latent capacity graph of Fig. 2.20 is traversed from bus 4 towards bus 3. The only other bus which can be reached from bus 4 without traversing a saturated branch in Fig. 2.20 is bus 5. Therefore, buses 4 and 5 are grouped in the first cluster, while buses 1, 2, and 3 are grouped into the second cluster. The branches which connect these two clusters are 1-5, 3-4 and 3-5. Therefore, the limiting critical cut-set is K_{crit} ={1-5,3-4,3-5}. This cut-set is said to be saturated by a margin of 30 MW for the outage of branch 3-4.

2.5.2 Application of the FT Algorithm on Different Network Flow Solutions

If the FT algorithm is applied to any of the three flow graphs (Fig. 2.13(a) or Fig. 2.13(b) or Fig. 2.13(c)) of the sample 5-bus power system to evaluate the outage of branch 3-4, the following conclusion will be reached independent of which flow graphs is used for the analysis: *the outage of branch 3-4* (e_4) *would maximally saturate cut-set* K_1 (where K_1={3-4,3-5, 1-5}) *by a margin of 30 MW*, i.e. $K_{crit} = K_1$ and $T_l = -30$ MW. This is verified as follows. Note that branch 3-4 is associated with 4 cut-sets as shown in Fig. 2.21. Fig. 2.22 depicts the power transfer across all four cut-sets associated with branch 3-4 for the flow graph obtained from a DC power flow solution. Consider cut-set K_3 in Fig. 2.22(c). The total power transfer across K_3 is 540 MW ($F_{K^3} = f_1 + f_2 + f_4 + f_7$) and the maximum power that can be transferred across K_3 after the outage of 3-4 is also 540 MW ($R_{K^3} = f_1^{max} + f_2^{max} + f_7^{max}$). As such, outage of 3-4 saturates K_3 by 0 MW ($R_{K^3} = F_{K^3}$). Similarly, it was observed that the outage of branch 3-4 creates a negative margin of 30 MW in cut-set K_1 (see Fig. 2.22(a)), positive margin of 40 MW in K_2 (see Fig. 2.22(b)), and a positive margin of 240 MW in K_4 (see Fig. 2.22(d)). *Therefore, it is validated using Fig. 2.22 that the FT algorithm correctly identifies the cut-set which gets saturated by the largest (negative) margin*.

Identical results were obtained when the FT was applied to the flow graphs of Fig. 2.13(b) and Fig. 2.13(c). Lastly, it must also be pointed out that the margin computed by the FT is indicative of the minimum amount of power transfer that must be reduced across cut-set K_{crit} to alleviate its saturation due to the contingency.

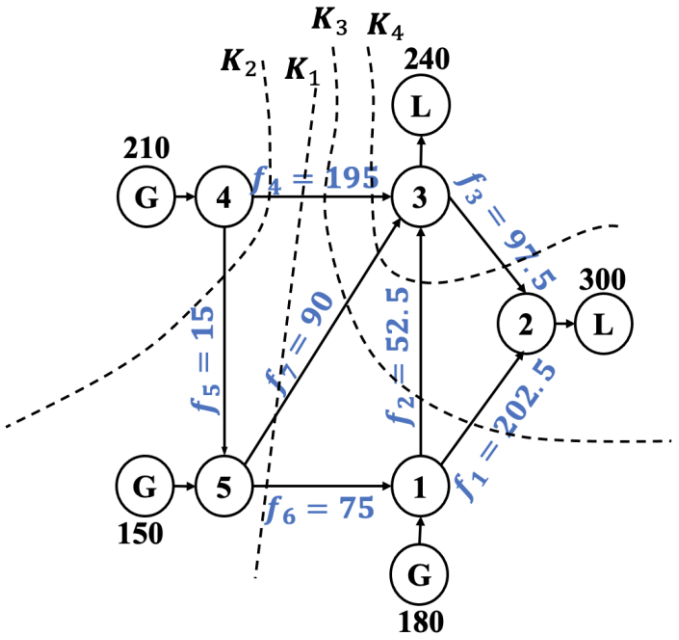

Fig. 2.21 The 4 Different Cut-sets Associated with Branch 3-4 (the Power Flows Correspond to a DC Power Flow Solution)

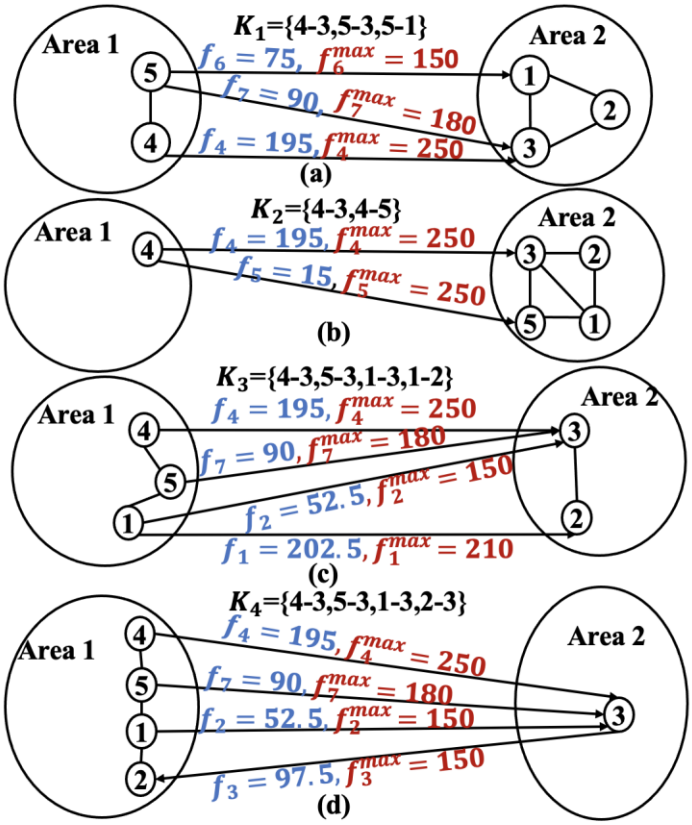

Fig. 2.22 Effect of the Outage of Branch 3-4 on (a) K_1, (b) K_2, (c) K_3, and (d) K_4 of the Flow Graph of Fig. 2.13(a).

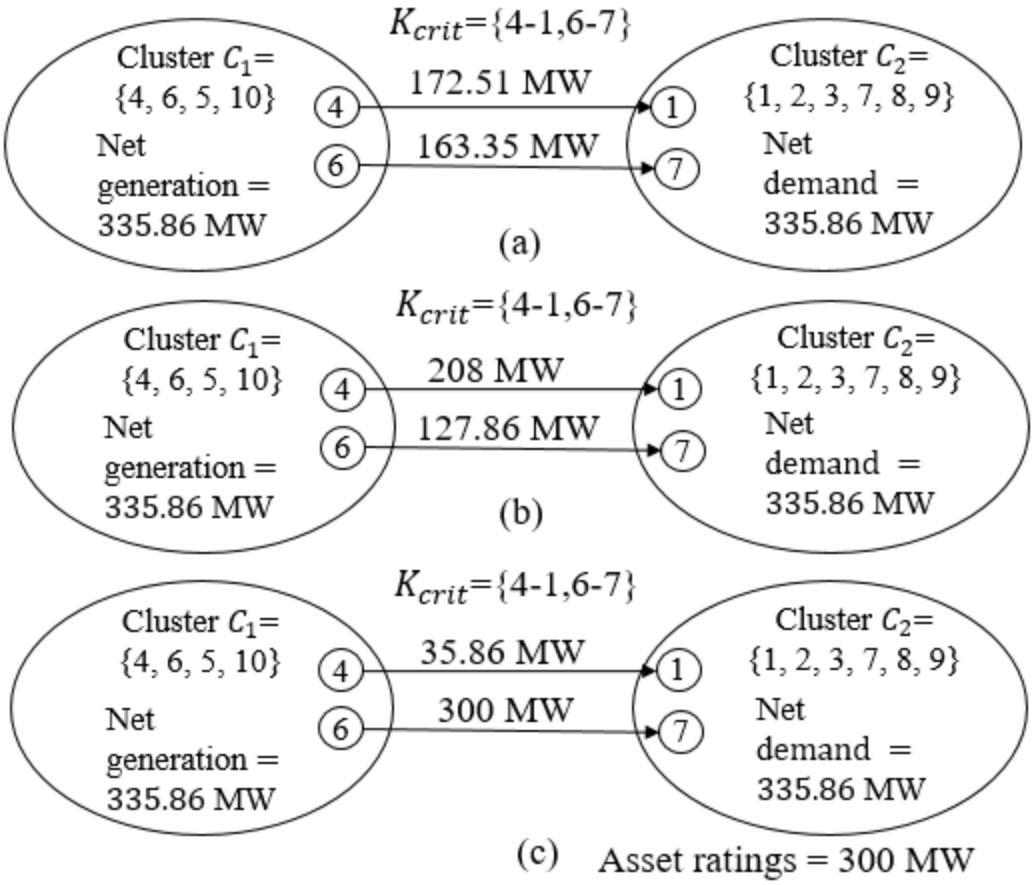

Fig. 2.23 (a) Power Transfer Across Cut-set K_{crit} for the Flow Graph of Fig. 2.14, (b) Power Transfer Across Cut-set K_{crit} for the Flow Graph of Fig. 2.15, (c) Power Transfer Across Cut-set K_{crit} for the Flow Graph of Fig. 2.16

Now, let us consider the sample 10-bus test system of Fig. 2.14. If branch 4-1 is evaluated by FT with respect to any of the flow graphs (Fig. 2.14 or Fig. 2.15 or Fig. 2.16), following observation is made: branch 4-1 is a special asset as it fails FT, and is associated with a limiting critical cut-set containing branches 4-1 and 6-7 (i.e., K_{crit} ={4-1,6-7}) with a transfer margin T_l =-35.86 MW. The implication of the above statement is explained with the help of Fig. 2.23(a), (b), and (c) which present the power transfer across cut-set K_{crit} for the three different flow graphs of the same system shown in Fig. 2.14, 2.15, and 2.16, respectively. From Fig. 2.23 it is clear that although the individual flows on different branches of the cut-set are different, FT finds that, for all three flow graphs, if the branch

4-1 is lost, the cut-set K_{crit} will have a power transfer capability shortage of 35.86 MW from cluster C_1 to cluster C_2. For example, in Fig. 2.23(a), when branch 4-1 is lost, the flow in branch 6-7 becomes (208+127.86) MW = 335.86 MW, which exceeds its rating (of 300 MW) by 35.86 MW. *In summary, the FT: (a) detects special assets, (b) identifies the limiting critical cut-set associated with each special asset, and (c) computes the power transfer margin across the identified limiting critical cut-set.*

2.6 Update Scheme (UPS) of the Network Flow Solution

During major power system disturbances, multiple outages can occur in rapid succession. Therefore, the FT results would also change following the outage of a branch. To identify the set of special assets following an outage, it is important to first update the graph theory-based network flows to account for the outage of any branch. The advantage of graph theory-based flows is that rerouting of the flow upon the loss of a branch can be achieved extremely fast. The technique of updating the flow graph $\mathcal{F}(V, E)$ and latent capacity graph $\mathcal{C}(V, E)$ when branch e_l suffers an outage is done in accordance with Algorithm III, which describes the graph theory-based update scheme (UPS).

2.6.1 Illustration of the UPS Algorithm

A graph theory-based network flow solution of the sample 5-bus system is presented in Fig. 2.24. We now want to find the updated flow and latent capacity graphs if branch 5-3 is lost. The steps involved in the UPS are explained below.

Algorithm III: Graph theory-based update scheme (UPS)

i. Let the flow to be rerouted be given by $F = f_l$, where f_l refers to the flow through branch e_l from bus v_l^F to v_l^T.

ii. Remove branch e_l from $\mathcal{F}(V, E)$ and $\mathcal{C}(V, E)$.

iii. Search $\mathcal{C}(V, E)$ to obtain the shortest unsaturated path $\mathcal{P}$ from v_l^F to v_l^T using breadth first search (BFS) [95].

iv. Find the maximum extra flow, $C_{\mathcal{P}}$, that can be rerouted through path $\mathcal{P}$.

v. If $F > C_{\mathcal{P}}$, inject $C_{\mathcal{P}}$ units of flow through path $\mathcal{P}$ and update F as $F \coloneqq F - C_{\mathcal{P}}$. If $F \leq C_{\mathcal{P}}$, inject F units of flow through path $\mathcal{P}$ and set $F \coloneqq 0$. Update the weights of $\mathcal{F}$ and $\mathcal{C}$ accordingly.

vi. Repeat Steps (ii) through (v) until $F = 0$.

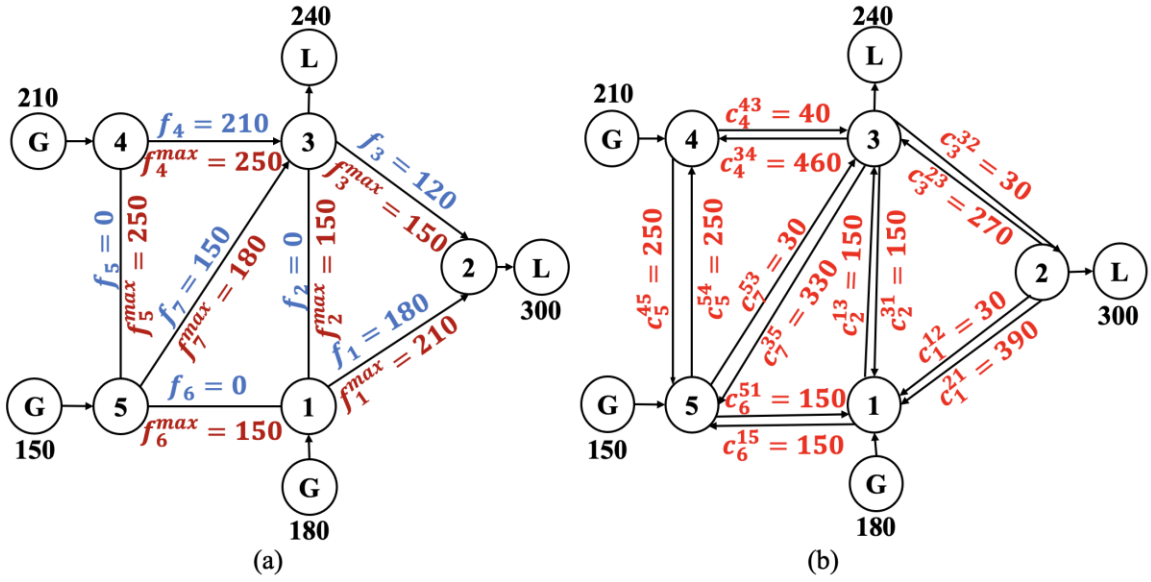

Fig. 2.24 (a) Flow Graph, and (b) Latent Capacity Graph of the 5-bus Power System Obtained from the Graph-theory Based Network Flow Solution

Step i: Since 150 MW of power flows through branch 5-3 from bus 5 towards bus 3 (see Fig. 2.11), the amount of power that must be rerouted when branch 5-3 is lost is $F = 150$ MW.

Step ii: The branch 5-3 is removed from both the flow graph and the latent capacity graph as shown in Fig. 2.25 below.

Step iii: The shortest unsaturated path from bus 5 to bus 3 is obtained using BFS. The path is given as follows: $\mathcal{P} = \{5 - 1 - 3\}$.

Step iv: The maximum power that could be rerouted through path $\mathcal{P}$ is 150 MW (i.e., $C_{\mathcal{P}} = 150$).

Step v: Since in this situation $F = C_{\mathcal{P}}$, 150 MW of power is rerouted through path $\mathcal{P}$. The flow and the latent capacity graphs are updated as shown in Fig. 2.25. Finally, F is updated as follows: $F = F - C_{\mathcal{P}} = 150 - 150 = 0$.

Step vi: Since $F = 0$, the steps (ii) through (v) are not repeated and the graphs shown in Fig. 2.25 are the updated flow and latent capacity graphs of the system. For large power systems it can be shown that the rerouting of the flow through the indirect paths actually occurs through a very small subgraph of the entire network. This will be discussed with the help of a case-study in the next Chapter.

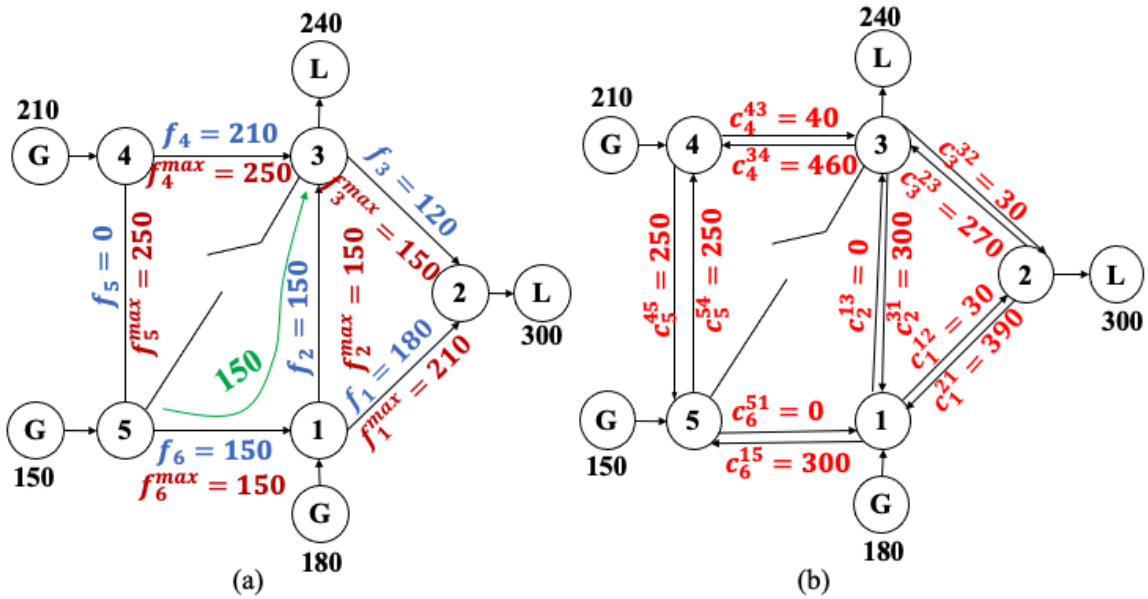

Fig. 2.25 (a) Flow Graph, and (b) Latent Capacity Graph After Power Flow Through Branch 5-3 is Re-routed Along Path 5-1-3 Using UPS Algorithm After Outage of Branch 5-3

We observed from the above example that the UPS algorithm creates an updated flow graph utilizing the set of shortest indirect paths to re-route the flows after a branch outage. This is possible because in the context of detecting saturated cut-sets, the net power transfer across any cut-set of the network is important, rather than the individual branch flows. This is explained with the help of another flow solution obtained from the DC power flow after the branch outage. Fig. 2.26(a) and (b) present the flow and latent capacity graphs obtained from a DC power flow after the outage of branch 5-3. Further Fig. 2.27(a) and (b) compares the flow graphs obtained from the UPS algorithm and the DC power flow solutions respectively. Despite the individual branch flows being different, the power transfer across any cut-set of the network remains the same (compare Fig. 2.27(a) and (b)). For example, the total power transfer across cut-set K_1 is 360 MW in both the graphs. Consequently, the FT uniquely determines post-contingency cut-set saturation (independent of whichever flow graphs are used for the network analysis). For instance, it detects that

outage of branch 4-3 will saturate cut-set K_1 by 210 MW. This is because the power transfer capacity of cut-set K_1 reduces to only 150 MW after the outage of branch 4-3.

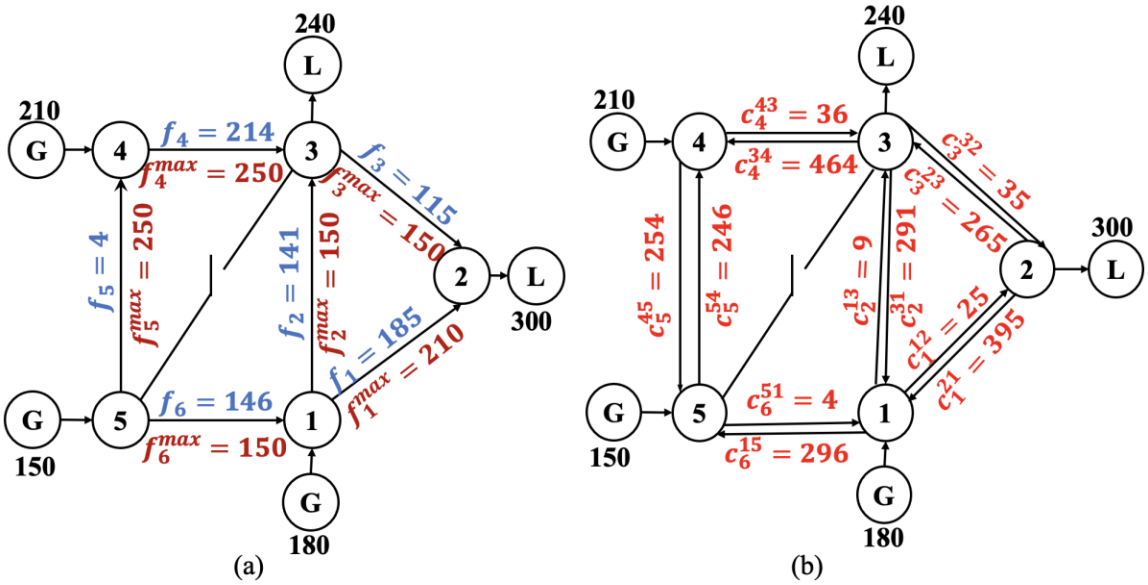

Fig. 2.26: (a) Updated Flow Graph, and (b) Updated Latent Capacity Graph Obtained from a DC Power Flow Solution After the Outage of Branch 5-3

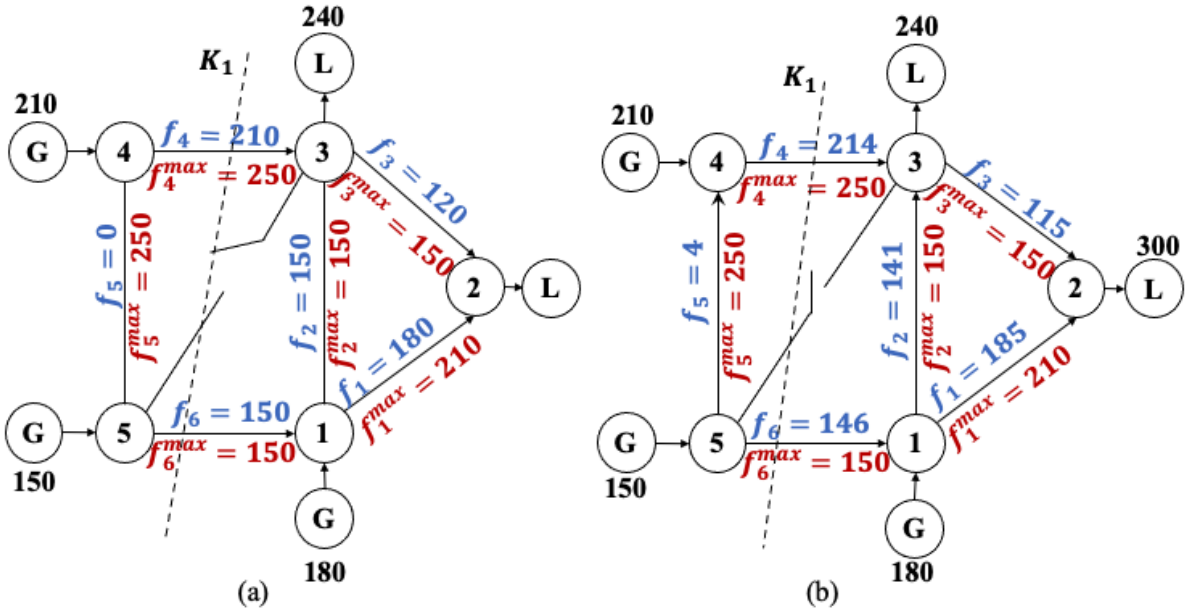

Fig. 2.27: (a) The Updated Flow Graphs Obtained from the UPS Algorithm, and (b) DC Power Flow Solution After the Outage of Branch 5-3

2.7 Shortlisting Assets (SA) Algorithm for Successive FT

In the base-case scenario when the flow graph is built for the first time all transmission assets would be investigated by the FT. However, in the event of a branch outage, when the UPS provides an updated flow graph, it is not necessary to evaluate all the assets by the FT once again to identify the special assets. By intelligently exploiting the information provided by FT in the base-case scenario and using the UPS to reroute the flow for the branch that is out, the FT can be performed on only a subset of the assets to evaluate the impact of a second contingency. This is explained through Fig. 2.28.

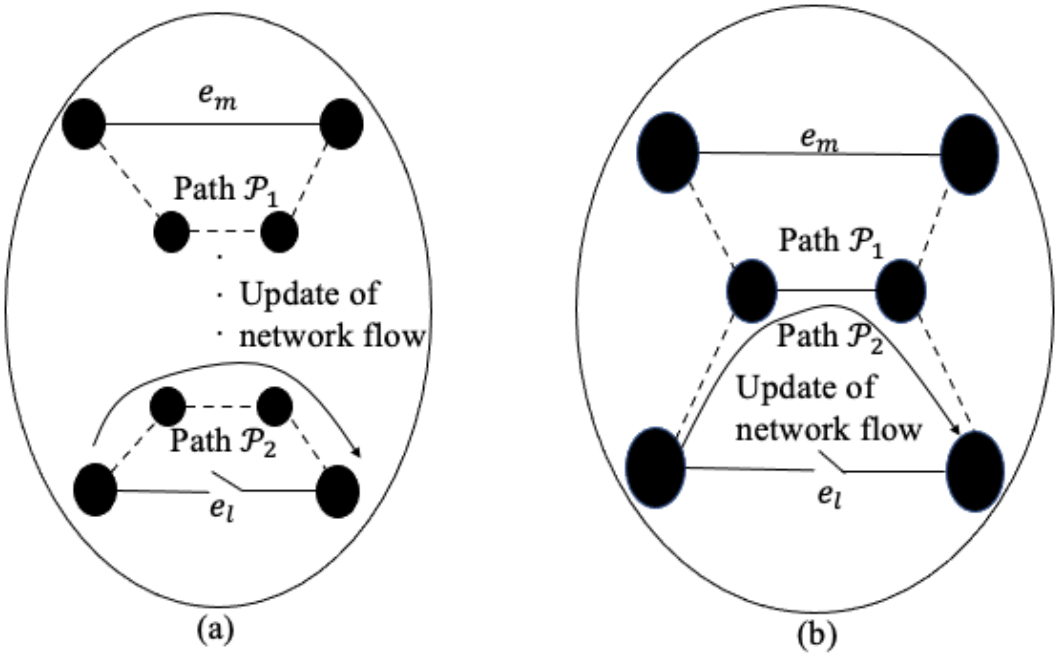

Fig. 2.28 (a) Rerouting the Flow on Branch e_l Does Not Involve any Branch of the Indirect Paths of e_m, and (b) Rerouting the Flow on Branch e_l Involves Some Branches of the Indirect Paths of e_m

Let it be known from the base-case FT that the flow through branch e_m can be rerouted through path $\mathcal{P}_1$, while the loss of branch e_l alters flow through path $\mathcal{P}_2$. Then, in Fig. 2.28(a), when branch e_l goes out, the flow through e_l is rerouted through $\mathcal{P}_2$ by the UPS. Now, since $\mathcal{P}_1$ and $\mathcal{P}_2$ do not involve common branches, the rerouting of power

through $\mathcal{P}_2$ by UPS does not necessarily modify the flows through $\mathcal{P}_1$; therefore, the FT need not be repeated for e_m. However, if $\mathcal{P}_1$ and $\mathcal{P}_2$ have common branches, as seen in Fig. 2.28(b); i.e., rerouting of the flow of branch e_l affects the flow through $\mathcal{P}_1$, then e_m must be evaluated by FT once again after the outage of e_l. This rationale of screening the assets to be evaluated by the FT in the event of an outage is called the shortlisting asset (SA) algorithm. The SA algorithm is explained with an example in the following section.

2.7.1 Illustration of the SA Algorithm

Let us consider a flow graph for a 7-bus power system shown in Fig. 2.29. The information obtained from the FT for every branch (in the base-case) is recorded in the form of a list as shown in Table 2.3. All branches passed the FT in the base-case scenario. Let us consider the FT result for the outage of branch 1-6. Table 2.3 indicates that 30 MW of power flowing through branch 1-6 can be rerouted through the path 1-3-6 by the UPS if the branch 1-6 is lost. Similarly, the 100 MW of power flowing through branch 3-1 can be rerouted through the indirect paths 3-6-1, 3-4-6-1, and 3-4-5-6-1, respectively, by the UPS, if the branch 3-1 is lost. The set of indirect paths for the other branches through which the power flows could be rerouted is given in the second column of Table 2.3.

Now, consider that the branch 2-1 suffers an outage at a particular time instant. By using the UPS algorithm, the flow through the branch 2-1 is rerouted through path $\mathcal{P} = \{2-3-6-1\}$, as shown in Fig. 2.30. The updated base-case network flow solution is obtained by the UPS. *Only specific transmission assets whose indirect paths involve branches 2-3, 3-6, and 6-1 must be re-evaluated by the FT*. From the second column of Table 2.3 it is observed that indirect paths of branches 5-6, 4-5, 4-7, 7-6, and 7-5, through

which rerouting can occur *do not involve* branches 2-3, 3-6, or 6-1. Therefore, the FT *need not be* performed for all the 11 branches. The FT *must be* performed for only six branches, namely, 1-6, 3-2, 3-1, 3-6, 4-3, and 4-6, whose indirect paths involve branches 2-3, 3-6, or 6-1. This is how the shortlisting of assets is achieved in a power network. By this rationale, even for a very large system, the FT needs to only evaluate a very small subset of assets, following a branch outage.

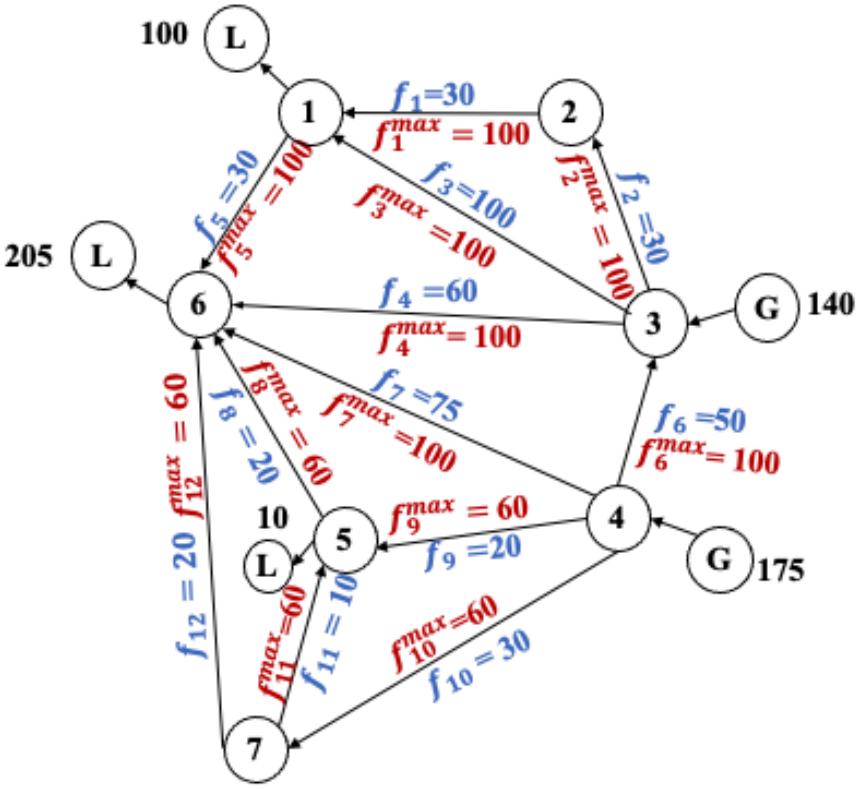

Fig. 2.29 A Flow Graph for a Sample 7-bus Power System

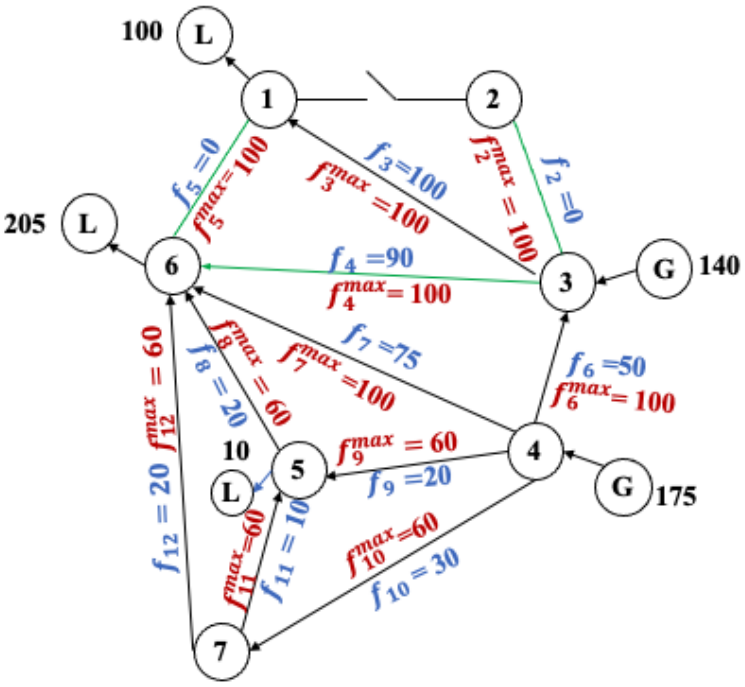

Fig. 2.30 With the Outage of Branch 2-1, the Flow is Re-routed Through Path $\mathcal{P}$={2-3-6-1}

Table 2.3: Information Recorded by the FT in the Base-case Scenario

Branch evaluated by FT	Indirect paths through which rerouting of flow can occur
Branch 1-6	Indirect Path 1: 1-3-6
Branch 2-1	Indirect Path 1: 2-3-6-1
Branch 3-2	Indirect Path 1: 3-6-1-2
Branch 3-1	Indirect Path 1: 3-6-1 Indirect Path 2: 3-4-6-1 Indirect Path 3: 3-4-5-6-1
Branch 3-6	Indirect Path 1: 3-4-6 Indirect Path 2: 3-4-5-6
Branch 4-3	Indirect Path 1: 4-6-3 Indirect Path 2: 4-5-6-3
Branch 4-6	Indirect Path 1: 4-3-6 Indirect Path 2: 4-5-6
Branch 5-6	Indirect Path 1: 5-4-6
Branch 4-5	Indirect Path 1: 4-6-5
Branch 4-7	Indirect Path 1: 4-5-7
Branch 7-6	Indirect Path 1: 7-5-6
Branch 7-5	Indirect Path 1: 7-4-5

CHAPTER 3

RESULTS: DETECTION OF SATURATED CUT-SETS

This Chapter presents the case-studies for different test systems. The proposed FT algorithm is applied on the IEEE 39-bus test system in the base-case scenario. The performance of the FT is compared with other contingency ranking methods on the IEEE 118-bus system, during successive outages. The scalability and computational efficiency of the FT algorithm is validated on the 17,941-bus model of the Western Interconnection. This Chapter also contains detailed discussions on the capabilities and limitations of the FT algorithm with a variety of examples and comparative studies.

3.1 Detection of Saturated Cut-sets in IEEE 39-bus System in Base-case

The system data for the IEEE 39-bus test system is obtained from MATPOWER [102]. When every transmission asset was investigated by the FT in base-case scenario, four saturated cut-sets were identified which are depicted by dotted lines in Fig. 3.1. The detailed information obtained from the graph theory-based FT is summarized below:

i. Outage of 11-10 saturates cut-set K_{crit}^{1}= {11-10,13-10} by 61 MW. Similarly, outage of 13-10 saturates the same cut-set K_{crit}^{1} by the same margin, because the branches 11-10 and 13-10 have the same rating of 600 MVA, and the total power transferred across cut-set K_{crit}^{1} is 661 MW.
ii. Outage of 6-11 saturates cut-set K_{crit}^{2}= {6-11,14-13} by 52 MW. However, outage of 14-13 saturates the same cut-set K_{crit}^{2} by 172 MW. This is because branches 6-11 and 14-13 have ratings of 480 MVA and 600 MVA, respectively, and the total power transferred across cut-set K_{crit}^{2} is 652 MW.

iii. Outage of 21-22 saturates cut-set K_{crit}^{3}= {21-22,24-23} by 393 MW, and the outage of 24-23 saturates K_{crit}^{3} by 93 MW. This is because branches 21-22 and 24-23 have ratings of 900 MVA and 600 MVA, respectively, and the total power transferred across cut-set K_{crit}^{3} is 993 MW.

iv. Outage of 16-21 saturates cut-set K_{crit}^{4}= {16-21,24-23} by 119 MW. Similarly, outage of 24-23 has the same effect on K_{crit}^{4}. This is because both lines have the same rating of 600 MVA, and the total power transferred across K_{crit}^{4} is 719 MW.

It is important to note that the proposed analysis not only identifies the saturated cut-sets, but also indicates the minimum amount of power transfer that must be reduced across the cut-set to alleviate its saturation. The performance of FT during *a series of outages* is studied in the next section.

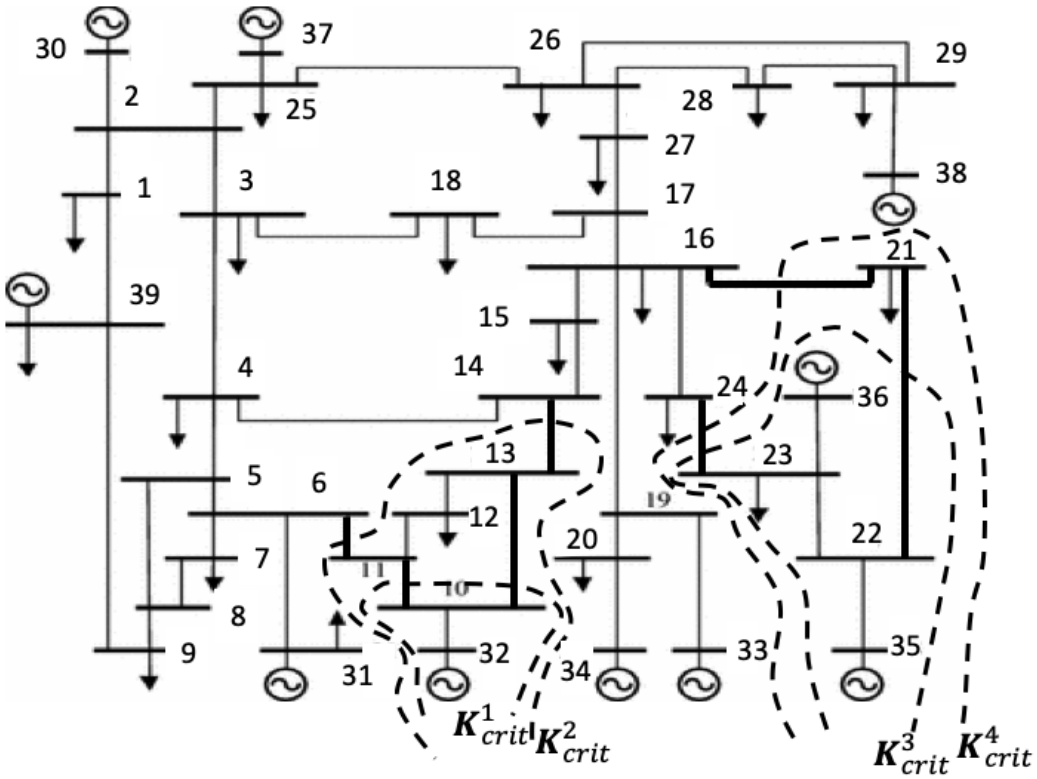

Fig. 3.1 Identification of Saturated Cut-sets in the IEEE 39-bus System for the Base-case Scenario

3.2 Detection of Saturated Cut-sets in IEEE 118-bus System During Outages

3.2.1 Performance of the FT Algorithm

The utility of the proposed FT algorithm for enhanced situational awareness is explained with a case-study on the IEEE 118-bus system. Due to a hurricane, let the following transmission asset outages occur one after another: 15-33, 19-34, 37-38, 49-66, and 47-69 (marked O_1 through O_5 in Fig. 3.2). From Fig. 3.2 and Table 3.1, following information is obtained when the algorithm is applied as outages manifest:

1) *Base-case:* In the base-case scenario, branch 26-30 fails the graph theory-based FT and is classified as a special asset. The loss of 26-30 would saturate the limiting critical cut-set K_{crit}^{0} by a margin of -77 MW, i.e., T_l^0= -77 MW.
2) *1st Outage:* When 15-33 is lost, no additional special assets are identified.
3) *2nd Outage:* When 19-34 is lost, no additional special assets are identified.
4) *3rd Outage:* When 37-38 is lost, the asset 42-49 fails the FT and is classified as a special asset. The loss of 42-49 would saturate the limiting critical cut-set K_{crit}^{3} by a margin of -186 MW, i.e., T_l^3= -186 MW.
5) *4th Outage:* When 49-66 is lost, no additional special assets are identified.
6) *5th Outage:* When 47-69 is lost, the assets 59-56, 63-59, 63-64, and 64-65 are classified as special assets. The loss of these four assets would saturate the limiting critical cut-sets, K_{crit}^{5a}, K_{crit}^{5b}, K_{crit}^{5c}, and K_{crit}^{5d}, by margins of -64, -191, -191, and -219 MW, respectively (i.e., T_l^{5a}= -64 MW, T_l^{5b}= -191 MW, T_l^{5c}= -191 MW, T_l^{5d}= -219 MW).

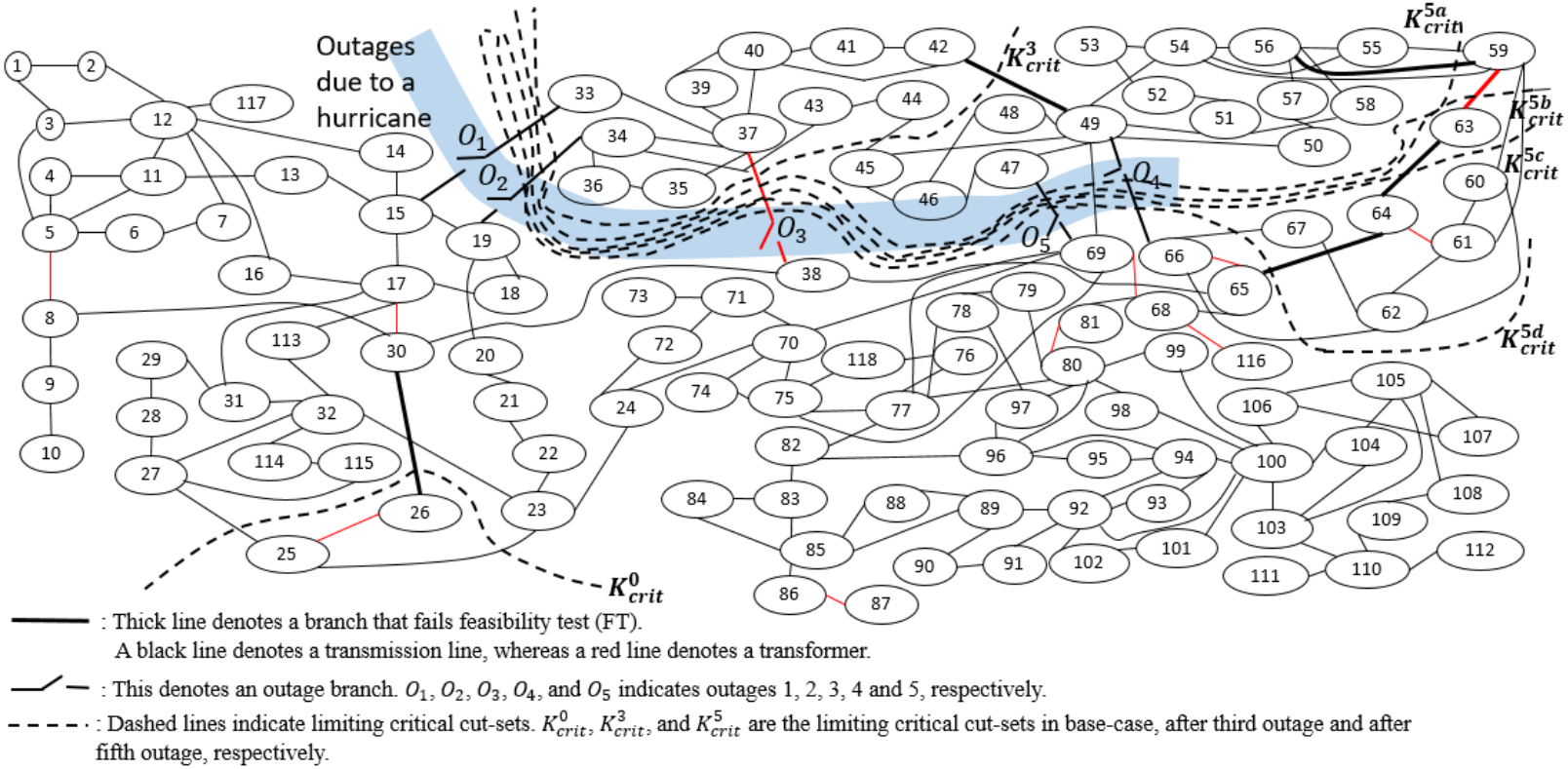

Fig. 3.2 Real-time Identification of Limiting Critical Cut-sets on the IEEE 118-bus Test System by the FT Algorithm During a Sequence of Outages

The value of the information obtained above can be realized by considering the following scenario: *after the occurrence of the fifth outage, the FT algorithm would inform the power system operators that if any of the four assets identified in the last row, second column of Table 3.1 is lost next (as the 6th outage), the corresponding cut-set identified in the third column would be saturated by the margin mentioned in the fourth column*. If this anticipated overload is to be avoided, the operator must preemptively reduce the power flowing through the identified cut-set by *at least* the amount mentioned in the last column of Table 3.1. Thus, the proposed network analysis tool is an enhanced power system connectivity monitoring scheme that improves the power system operators' situational awareness by augmenting their visualization in real-time. Also, it must be noted that this insight is very different from what a traditional contingency analysis scheme may provide.

Table 3.1: Identification of Limiting Critical Cut-sets in IEEE 118-bus Test System

Event	New Special Asset	Limiting Critical Cut-set	Transfer margin (MW)
Base-case	26-30 (345 kV line)	K_{crit}^{0} = {26-30,25-27,25-23}	T_l^0=-77
Outage 1 (15-33)	-	-	-
Outage 2 (19-34)	-	-	-
Outage 3 (37-38)	42-49 (138 kV line)	K_{crit}^{3}= {42-49,44-45}	T_l^3= -186
Outage 4 (49-66)	-	-	-
Outage 5 (47-69)	59-56 (138 kV line)	K_{crit}^{5a}= {59-56,59-54,59-55,69-49}	T_l^{5a}= -64
	63-59 (345/138 kV transformer)	K_{crit}^{5b}= {63-59,61-59,60-59,69-49}	T_l^{5b}= -191
	63-64 (345 kV line)	K_{crit}^{5c}= {63-64,61-59,60-59,69-49}	T_l^{5c}= -191
	64-65 (345 kV line)	K_{crit}^{5d}= {64-65,66-62,66-67,69-49}	T_l^{5d}= -219

3.2.2 Comparative Analysis with Different Methods

This section provides a brief review of two other contingency ranking techniques proposed in prior literature. Subsequently, the FT is compared with these contingency ranking techniques. Moreover, the results from FT are validated using an independent cascading simulation analysis.

3.2.2.1 Contingency Ranking Using PTDFs

The power transfer capacity from a source (generator) bus v_i to a sink (load) bus v_j is as follows [53]:

$$C_i^j = Min\left\{\frac{f_1^{max}}{|PTDF_{1,i}^j|}, \dots, \frac{f_l^{max}}{|PTDF_{l,i}^j|}, \dots, \frac{f_m^{max}}{|PTDF_{m,i}^j|}\right\} \tag{3.1}$$

where, f_l^{max} denotes the asset ratings, $PTDF_{l,i}^j$ denotes the power transfer distribution factor for a power injection at bus i and power withdrawal at bus j, and m denotes total number of transmission assets. Now, [53] defines the electrical betweenness for a potential branch contingency e_k as follows:

$$\mathcal{T}_k = \max\left[\mathcal{T}_k^p \middle| \mathcal{T}_k^n\right], \tag{3.2}$$

where, $\mathcal{T}_k$ denotes the electrical betweenness for a branch contingency e_k. $\mathcal{T}_k^p$ and $\mathcal{T}_k^n$ represent the positive and negative electrical betweenness of the branch e_k, which are obtained as follows:

$$\mathcal{T}_k^p = \sum_{\forall v_i \in G} \sum_{\forall v_j (v_j \neq v_i) \in L} C_i^j PTDF_{l,i}^j, \qquad if\ PTDF_{l,i}^j > 0 \tag{3.3}$$

$$\mathcal{T}_k^n = \sum_{\forall v_i \in G} \sum_{\forall v_j (v_j \neq v_i) \in L} C_i^j PTDF_{l,i}^j, \qquad if\ PTDF_{l,i}^j < 0 \tag{3.4}$$

The electrical betweenness $\mathcal{T}_k$ presented in (3.2) can be used to rank different transmission contingencies [53].

3.2.2.2 Contingency Ranking Using LODFs

Contingency ranking by LODFs was proposed in [55]. A metric called the line outage impact factor (LOIF) was computed using the LODF matrix. For a single branch contingency e_k, the impact of the contingency on all other transmission assets is quantified as follows:

$$LOIF_k = \sum_{\forall e_l \in E} LODF_{l,k} \tag{3.5}$$

where, $LOIF_k$ denotes the LOIF for a potential branch contingency e_k. LOIF can be used to perform contingency ranking [55].

3.2.2.3 Cascading Simulation Analysis Using MATCASC

MATCASC is an open-source MATLAB based tool, that evaluates the consequence of cascading failures in power systems due to branch overloads [76]. The use of MATCASC for cascade failure analysis is explained here with the help of a flowchart in Fig. 3.3(a). Any branch outage is considered an input to MATCASC as an initial triggering contingency. Following this initial outage, it solves DC power flows to check for overloads beyond the emergency rating of transmission lines. The transmission lines that have overloads beyond the emergency rating are tripped following which a DC power flow is solved again. The steps are repeated unless there are no successive overloads in the system. At the end of the cascade, the power system might have already been split into multiple islands due to the branch outages at different stages of the cascade, as shown in Fig. 3.3(b). If the total power supplied is greater than the power demand of an island, there is no unsatisfied

power demand in the island. On the other hand, if the total power supply is less than the total demand, then a fraction of the power is not satisfied in the island. Therefore, to find the contingencies that will trigger a cascade and result in unsatisfied power demand, every possible initial triggering branch outage is evaluated in MATCASC. Additional details of MATCASC could be found in [76].

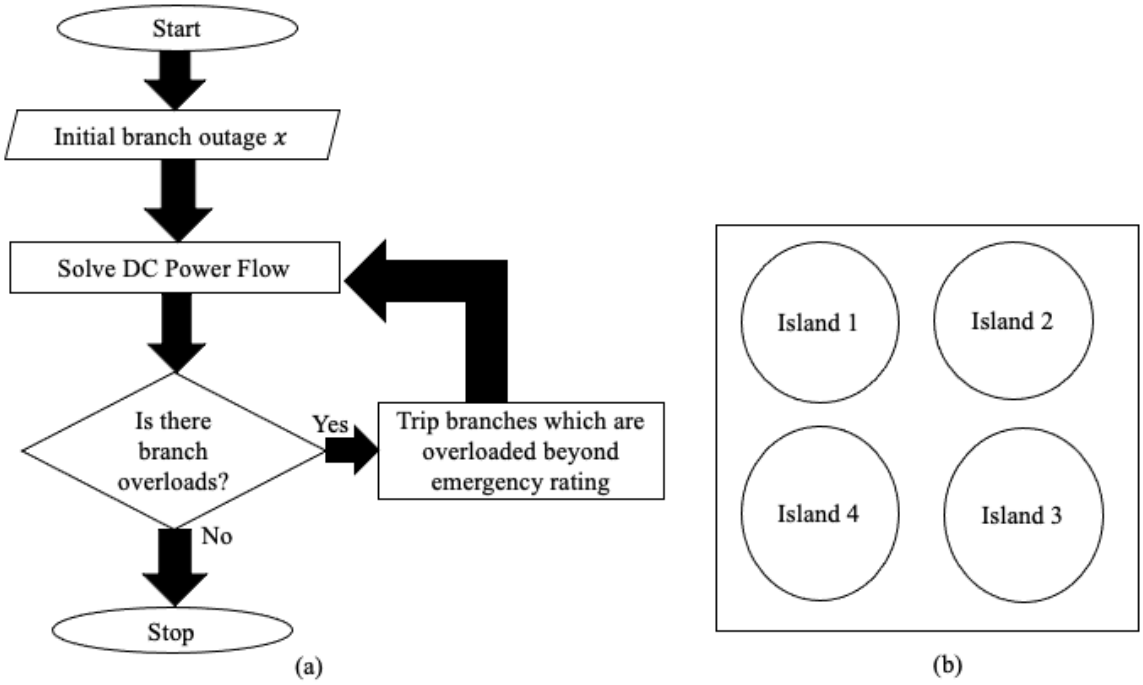

Fig. 3.3 (a) A Simplified Flowchart Showing how MATCASC Performs Cascading Failure Analysis for Any Initial Branch Outage, and (b) Formation of Different Islands at the End of the Cascade

3.2.2.4 Comparative Study on the IEEE 118-bus Test System

The output of the proposed FT algorithm is compared with those obtained from two power system vulnerability assessment techniques, namely, the metrics developed in [53] and [55] (described in the sub-sections 3.2.2.1 and 3.2.2.2, respectively). The analysis was performed on the IEEE 118-bus system for the same sequence of outages that were described in Table 3.1. Further, in order to validate the severity of different contingencies

identified by the FT, an independent cascading failure simulation was run in MATCASC (discussed in sub-section 3.2.2.3). The amount of load shed at the end of the cascade indicates the severity of the contingency. The results of the comparison are shown in Table 3.2.

Table 3.2 Ranking of Contingencies and Cascading Analysis in IEEE 118-bus Test System After Different Outages

Event	Cascading Analysis		Rank by [53]	Rank by [55]
	New contingency	Load shed		
Base-case	26-30 (345 kV line)	12.20%	20	42
Outage 1 (15-33)	-	-	-	-
Outage 2 (19-34)	-	-	-	-
Outage 3 (37-38)	42-49 (138 kV line)	29.87%	16	58
Outage 4 (49-66)	-	-	-	-
Outage 5 (47-69)	64-65 (345 kV line)	28.92%	6	167
	63-59 (345/138 kV transformer)	28.26%	8	70
	63-64 (345 kV line)	28.26%	9	73
	56-59 (138 kV line)	25.27%	15	119

Column 2 of Table 3.2 shows the contingencies identified by MATCASC that result in load shed as different events manifest in the IEEE 118-bus system. The ranking of these load-shed-causing-contingencies, obtained by the techniques developed in [53] and [55] are provided in Columns 4 and 5, respectively. It can be observed from Table 3.2 that the contingencies that result in loss of load were not the top ranked contingencies identified by the metrics developed in [53] and [55]. For instance, after the fifth outage, if any of the four new contingencies identified in Column 2 were to occur (as the sixth outage), then it would result in load shedding in excess of 25%. However, none of these four high load-

shed-causing-contingencies appeared in the top four ranked contingencies of [53] or [55]. On the other hand, all the load-shed-causing-contingencies were detected as special assets by the proposed FT algorithm (compare Column 2 of Table 3.2 with Column 2 of Table 3.1). This shows the usefulness of the FT in detecting critical contingencies.

3.2.3 Application of the FT Considering Different Asset Ratings

In the Section 3.2.2 the detailed performance of FT considering normal (or continuous) transmission asset (line or transformer) ratings were presented for the IEEE 118-bus test system. However, the power carrying capacities of transmission lines are influenced by several factors such as the air temperature, solar radiation, wind magnitude and wind direction, etc. [103]-[106]. The proposed FT algorithm is generic enough to detect saturated cut-sets based upon asset ratings determined by any criterion. To demonstrate this, we present the application of the FT for two different scenarios in the IEEE 118-bus test system: (a) *Scenario 1*: asset ratings with 95% of the normal value, and (b) *Scenario 2*: asset ratings with 105% of the normal value.

Column 1 of Table 3.3 lists the sequential outages. Columns two through four present the results of Scenario 1, whereas columns five through seven present the results corresponding to that of Scenario 2. Comparing the FT results for the two scenarios after different outages we observe that the violations detected by the FT algorithm are more severe for Scenario 1 as compared to Scenario 2 (because of more conservative asset ratings used in the former). For instance, the number of special assets identified by the FT is more in Scenario 1 than in Scenario 2 (compare the second and fifth columns of Table 3.3). Further, the transfer margin for the outage of a special asset on the respective limiting critical cut-

set is higher for Scenario 1 as compared to Scenario 2 (compare the fourth and seventh columns of Table 3.3).

Table 3.3: Performance of the FT Considering Different Transmission Asset Ratings During Multiple Outages

Event	Scenario 1: Rating: 95%×Normal			Scenario 2: Rating: 105%×Normal		
	New Special Asset	Limiting Critical Cut-set	Transfer margin (MW)	New Special Asset	Limiting Critical Cut-set	Transfer margin (MW)
Base-case	26-30	{26-30,25-27,25-23}	-99	-	-	-
Outage 1: (15-33)	-	-	-	-	-	-
Outage 2: (19-34)	-	-	-	-	-	-
Outage 3: (37-38)	42-49	{42-49, 44-45}	-197	42-49	{42-49, 44-45}	-175
Outage 4: (49-66)	63-59	{63-59,61-59, 60-59, 69-49, 47-69}	-15	-	-	-
	63-64	{63-64,61-59, 60-59, 69-49, 47-69}	-15			
	64-65	{64-65,62-66, 66-67, 49-69,47-69}	-43			
Outage 5: (47-69)	63-59	{63-59,61-59, 60-59, 69-49}	-224	63-59	{63-59,61-59, 60-59, 69-49}	-158
	63-64	{63-64,61-59, 60-59, 69-49}	-224	63-64	{63-64,61-59, 60-59, 69-49}	-158
	64-65	{64-65,62-66, 66-67, 49-69}	-252	64-65	{64-65,62-66, 66-67, 49-69}	-186
	56-59	{59-56, 59-54, 59-55, 69-49}	-97	56-59	{59-56, 59-54, 59-55, 69-49}	-31
	59-60	{60-59,61-59, 63-59, 69-49}	-6			
	59-61	{61-59,60-59, 63-59, 69-49}	-6			
	49-69	{69-49,61-59, 63-59, 60-59}	-6			

3.3 Time Comparisons of FT and RTCA on Different Test Systems

In this section, a statistical comparison of the computation time of the FT algorithm (after an outage) and traditional RTCA is presented for test systems of varying size (IEEE 118-bus, Texas 2000-bus, Polish 3375-bus, and the 9241-bus European transmission systems). Both the FT and RTCA were implemented in MATLAB on the same computer (Core i7, 3.60 GHz CPU processor with 16 GB RAM). For each test system the computation time of the FT and RTCA was monitored for different transmission outages (top 100 of the highest loaded transmission assets were considered as possible contingencies). Fig. 3.4 compares the computation time of the FT and RTCA for the four specified test systems. It can be clearly observed that the FT is at least an order of magnitude faster than an exhaustive RTCA. Further, it can be observed that the FT takes slightly less time for the 3375-bus system as compared to the 2000-bus system. This happens because the computation time of FT not only depends on the system size, but also the topological structure of the network and the current operating condition of the system.

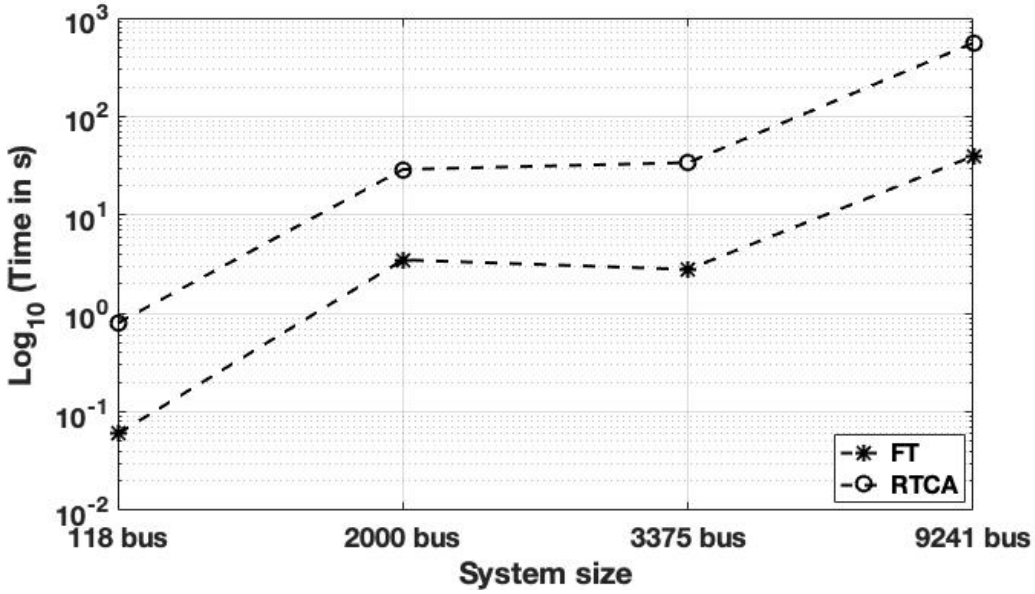

Fig. 3.4 Comparative Analysis of the Computation Time of the FT and RTCA for Test Systems of Different Sizes

3.4 Application of FT on a 17,941-bus Model of Western Interconnection

The proposed FT algorithm is applied on a 17,941-bus model of the Western Interconnection to test the scalability and computational speed of the proposed network analysis scheme. Sub-section 3.4.1 presents some statistics of graph theory-based FT and UPS which highlight the computational advantage of the proposed methodology. Sub-section 3.4.2 describes how the proposed network analysis scheme provides useful information when a sequence of outages occurs in this system.

3.4.1 Computational Efficiency of the Graph-theory Based Network Analysis

It takes 6 min to run an exhaustive *N*-1 FT for this system in the base-case scenario on a computer with Core i7, 3.60 GHz CPU processor and 16 GB RAM. When FT evaluates branch e_l for an outage, the indirect paths of e_l are traversed by BFS. However, the saturation of the set of indirect paths may occur when a small number of indirect paths are traversed by the graph theory-based FT. Moreover, since BFS always identifies the shortest path from the source to the sink, the number of branches contained in an indirect path would be relatively small. For every non-radial branch of this system, the number of indirect paths required to saturate the graph and the maximum number of branches contained in an indirect path is computed. The statistics of the FT algorithm is summarized in Fig. 3.5(a) and Fig. 3.5(b).

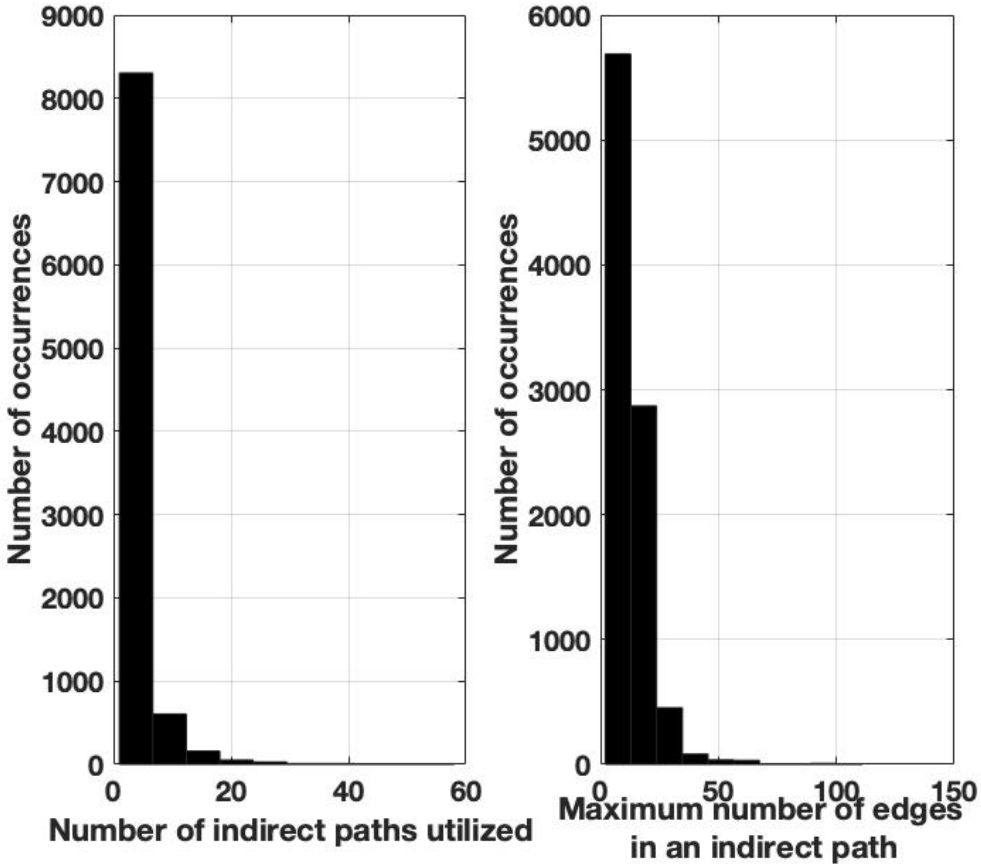

Fig. 3.5 (a) Histogram of Number of Indirect Paths Traversed by the Graph Theory-Based FT, and (b) Histogram of Maximum Number of Branches Contained in an Indirect Path

Fig. 3.5(a) plots the histogram for the number of indirect paths utilized by the BFS to saturate the latent capacity graph. The largest number of indirect paths required was 58. Fig. 3.5(b) plots the histogram of maximum number of branches contained in an indirect path traced by the BFS; the maximum was 111. Thus, the histogram plots demonstrate that the graph theory-based FT essentially uses a small subgraph to detect post-contingency cut-set saturation; this is the fundamental reason why the graph theory-based FT is computationally efficient. Moreover, it is important to note that in the base-case scenario all transmission assets were evaluated by the FT. But during a sequence of outages only a shortlisted number of transmission assets will be evaluated by the FT (utilizing the UPS and SA algorithms), which will further increase the computation speed.

3.4.2 A Case-study During a Series of Outages on Western Interconnection

This sub-section demonstrates the usefulness and scalability of the proposed approach through a *N*-1-1 event analysis of this system. The loss of 500 kV Hassayampa-North Gila (H-NG) transmission line was the first event, while the second event was the loss of 230/92 kV Coachella Valley transformers. Before the analysis was done for the outage of the events, it took approximately 0.5 s to build the flow graph and the latent capacity graph for the base-case. As mentioned earlier, it takes approximately 6 min to run FT on all transmission assets in the base-case. Whether events 1 and 2 resulted in any additional special asset was investigated as follows:

Event 1: Once the 500 kV H-NG transmission line was lost, graph theory-based UPS took only 0.20 s to reroute the flow to obtain a new flow graph. The SA scheme took 0.06 s to identify 271 branches that were to be examined by FT for this new graph. Time required by FT to examine all the 271 branches for an outage was 32 s. Among the 271 branches, 4 branches failed FT and were classified as special assets as shown in Table 3.4. For the 4 special assets, the FT found the corresponding limiting critical cut-set, K_{crit}; $|K_{crit}|$ in Table 3.4 denotes the number of branches contained in K_{crit}. Moreover, FT provided information regarding the impact of the loss of a special asset on the associated limiting critical cut-set. For example, if the transmission corridor 936-1192 is lost next, the limiting critical cut-set would be saturated by a margin of 441 MW. The total time required to perform this network analysis and identify all the limiting critical cut-sets after the outage of H-NG was 32.26 s (i.e., total time taken by UPS, SA, and FT). On the other hand, if FT were to be run on all transmission assets (as was done in the base-case), the time required

would be 6 min. Therefore, intelligently performing FT on a shortlisted set of transmission assets reduced the computation time from 6 min to 32.26 s.

Table 3.4 Application of Graph-Theory Based Network Analysis in the Western Interconnection

Events	Time of UPS	SA for FT		FT on shortlisted assets				Total Time
		#Branch	Time	New special assets	$\|K_{crit}\|$	T_l (MW)	Time	
Line outage: Hassayampa-North Gila	0.20 s	271	0.06 s	936-1192 (500 kV line)	57	-441	32 s	(0.20+0.06+32) =32.26 s
				1192-1217 (500 kV line)	49	-1258		
				2873-2902 (500 kV line)	18	-419		
				2902-2903 (500/230 kV transformer)	21	-309		
Transformer outage: Coachella Valley	0.06 s	82	0.07 s	2416-2488 (92 kV line)	8	-35.35	10 s	(0.06+0.07+10) = 10.13 s
				2421-2487 (230 kV line)	2	-2		
				2421-3293 (230 kV line)	2	-2		
				2438-2606 (230 kV line)	5	-55		
				2487-2488 (230/90 kV transformer)	8	-35		
				2712-2878 (230 kV line)	9	-35		

Event 2: When 230/92 kV Coachella Valley transformers are tripped, the UPS took only 0.06 s to obtain the updated network flow solution. Time required by the SA scheme to shortlist the branches to be evaluated by FT was 0.07 s; 82 new branches were shortlisted. Time required by FT to examine all the 82 shortlisted branches was 10 s. Among the 82 branches examined, 10 branches failed FT and were classified as special assets (see Table 3.4). Total time required to identify the set of special assets after the outage of Coachella

Valley transformers was 10.13 s. Therefore, it is again observed that the use of UPS and SA reduces the time required by the FT analysis after an outage.

3.5 Practical Utility of the FT Algorithm

After the 2011 U.S. Southwest blackout, the FERC reported the following finding [36]: "Affected TOPs (transmission operators) have limited visibility outside their systems, typically monitoring only one external bus. As a result, they lack adequate situational awareness of external contingencies that could impact their systems. They also may not fully understand how internal contingencies could affect SOLs (system operating limits) in their neighbors' systems." The recommendation of FERC to TOPs was to "review their real-time monitoring tools, such as state estimator and RTCA, to ensure that such tools represent critical facilities needed for the reliable operation of BPS (bulk power system)".

Now, modeling all "critical facilities" over a large area (across different utilities) could significantly increase the number of contingencies to be evaluated by RTCA, which would then increase the solution time considerably [40], [46]. In this regard, the *ability of the proposed network analysis to analyze the effects of any outage on very large systems and provide meaningful quantifiable information in a matter of seconds gives it a distinct advantage*. Moreover, the special assets detected by the FT can be suitable candidates for detailed analysis by a more precise CA tool. Thus, the proposed research can complement real-time operations by extending an operator's visibility to external contingencies, while alleviating the associated computational burdens.

3.6 The Limitation and Contribution of the FT algorithm

3.6.1 FT is not Guaranteed to Detect all Post-contingency Branch Overloads

As per the FT when all the indirect paths do not have sufficient capacity to reroute the power flowing through a branch, it implies that it would inevitably result in post-contingency branch overloads. However, the converse is not true. This is illustrated using the test system shown in Fig. 3.6, and the corresponding flows shown in Fig. 3.7 and Fig. 3.8.

Fig. 3.7(a) presents a DC power flow solution, when 100 MW of power is injected at bus 1, and 100 MW is withdrawn at bus 2 (Scenario 1). The numbers in non-bold fonts indicate flows, while the numbers in bold font denote ratings. The proposed FT algorithm identifies branch 1-2 as a special asset because the indirect paths of branch 1-2 do not have sufficient capacity to reroute the flow through the direct path, namely, branch 1-2. A post-contingency DC power flow shown in Fig. 3.7(b) validates that such an outage results in overloads along Indirect path 1.

Fig. 3.8(a) presents a DC power flow solution, when 85 MW of power is injected at bus 1, and the same is withdrawn at bus 2 (Scenario 2). In this scenario, the proposed FT algorithm does not identify branch 1-2 as a special asset because the set of indirect paths have sufficient capacity to reroute the flow of the direct path. However, a post-contingency DC power flow solution shown in Fig. 3.8(b) indicates that the Indirect path 1 is still overloaded, due to lower impedance of Indirect path 1 compared to Indirect path 2.

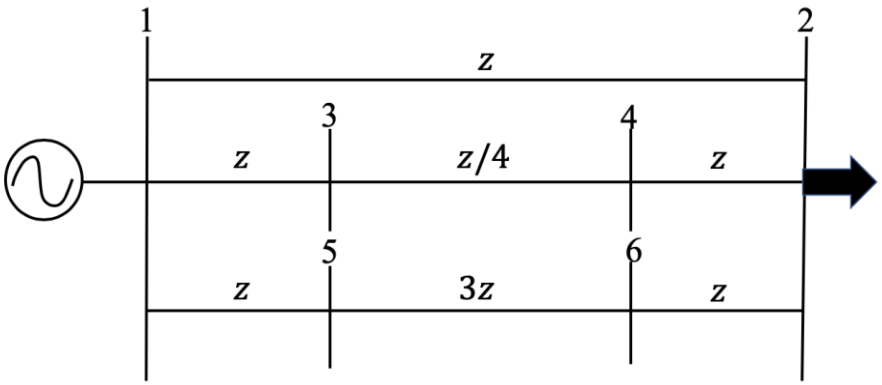

Fig. 3.6 Topology of a Sample 6-bus Power System (Branch Impedances are Represented in Terms of a Variable *z*)

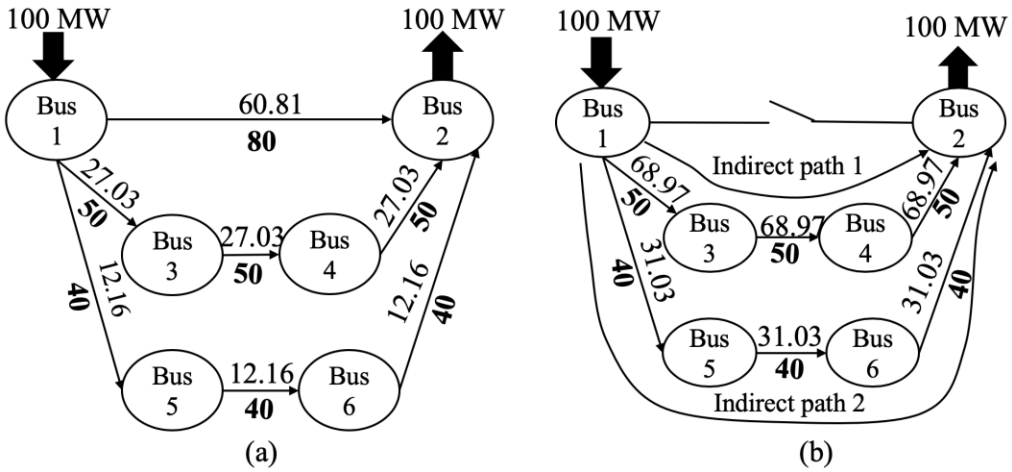

Fig. 3.7 Scenario 1-(a) A DC Power Flow Solution in Base-case, and (b) A DC Power Flow Solution for the Outage of Branch 1-2

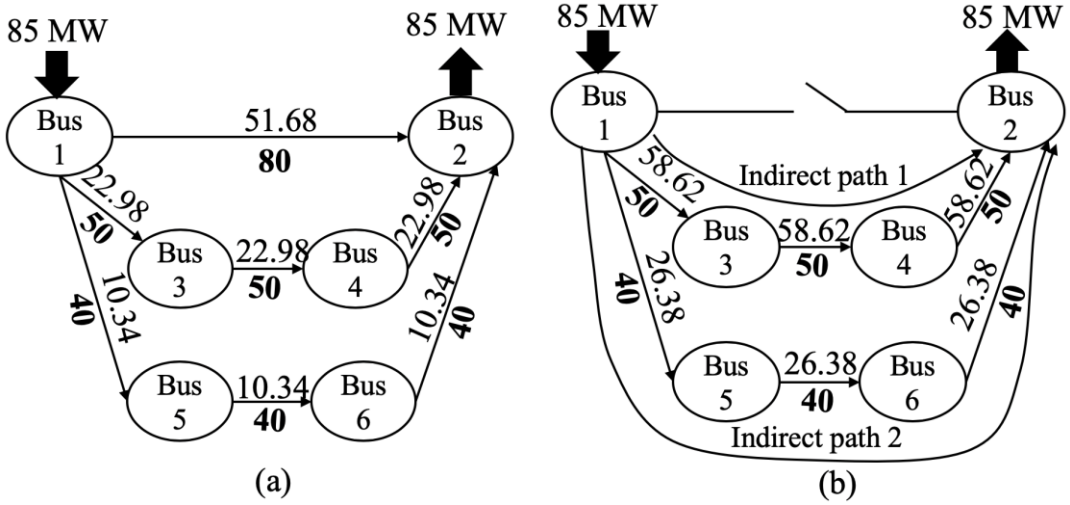

Fig. 3.8 Scenario 2-(a) A DC Power Flow Solution in Base-case, and (b) A DC Power Flow Solution for the Outage of Branch 1-2

From this illustration, the following conclusions can be drawn: when the set of indirect paths do not have the capacity to reroute the power flowing through the direct path (see Fig. 3.7), no additional information is required to conclude that there would be a post-contingency overload. The FT takes advantage of this observation to identify violations quickly. At the same time, the FT is not able to capture the overload occurring in Fig. 3.8. This is because the graph theory-based network flow algorithm ignores the effects of impedances when creating the flows. Thus, the proposed approach may not detect all possible post-contingency branch overloads.

3.6.2 FT is Guaranteed to Detect all Post-contingency Cut-set Saturation

The discussion presented in Section 3.6.1 reveals that the graph theory-based FT is not guaranteed to identify all contingencies that create post-contingency branch overloads. However, the FT *does guarantee* detection of all contingencies that create a saturated cut-set in the network. This is explained as follows. Let us examine if the outage of branch e_l of Fig. 3.9 would create a saturated cut-set in the system using the proposed FT. Branch e_l could be associated with multiple cut-sets in the system. With reference to Fig. 3.9 the i^{th} cut-set associated with branch e_l is denoted as follows:

$$K_i = \left\{e_l, e_{l_1}, e_{l_2}, \dots, e_{l_{(k-1)}}\right\} \quad \text{for } 1 \leq i \leq x \tag{3.6}$$

where, k is the total number of branches in cut-set K_i, and x is the total number of cut-sets associated with branch e_l. When the transfer margin, T_l, computed by the FT is negative it implies that the outage of branch e_l saturates at least one cut-set, among the x cut-sets that branch e_l is associated with. On the other hand, if the transfer margin, T_l, computed by the FT is positive, it implies that the outage of branch e_l does not saturate any of the x cut-sets

that it is associated with. Therefore, the FT will not miss a single contingency that would create a saturated cut-set. This is further illustrated using the test system shown in Fig. 3.10, and the corresponding flows shown in Fig. 3.11 and Fig. 3.12.

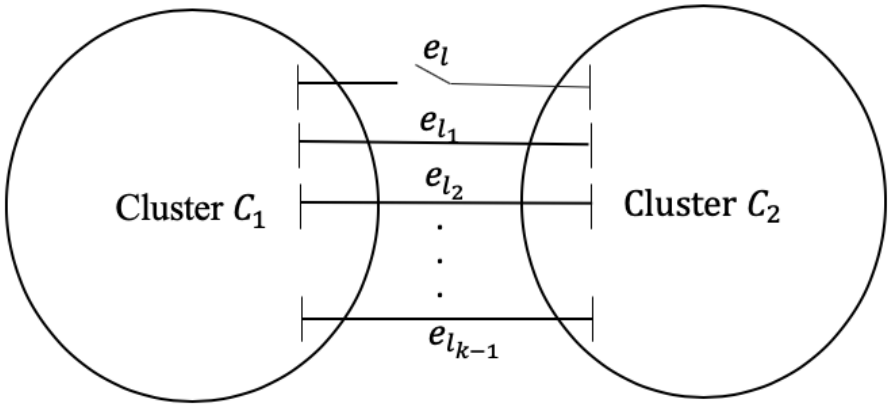

Fig. 3.9 K_i is the i^{th} Cut-set (Among x Cut-sets) Associated with Branch e_l that Separates the Network into Two Disjoint Clusters

Fig. 3.11 presents a DC power flow solution when the total load and generation in the system is 594 MW (Case 1). The FT algorithm finds that the outage of 3-4 saturates cut-set K_2 ={3-4,3-5,1-5} by 31 MW. To validate this inference, the power transfer capability across each cut-set associated with branch 3-4 is enumerated from the DC power flow solution. As shown in Fig. 3.11, branch 3-4 is associated with four cut-sets: K_1,K_2, K_3, and K_4. The power transfer capabilities across the four cut-sets of the test system when branch 3-4 is lost are summarized in Table 3.5, where P_K, denotes the total flow that is to be transferred across the cut-set, and R_K denotes the total capacity of all the branches belonging to the cut-set (excluding branch 3-4 itself). It is observed that P_K is greater than R_K only for cut-set K_2 by 31 MW. This verifies that for Case 1, the outage of branch 3-4 would saturate cut-set K_2 by 31 MW.

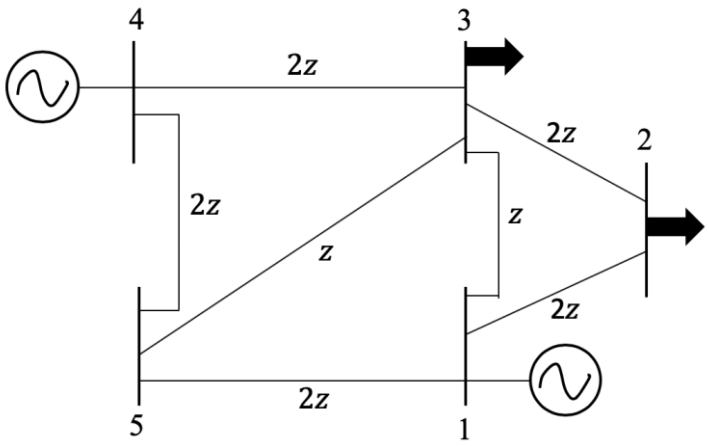

Fig. 3.10 Topology of a Sample Five-bus Power System (Branch Impedances are Represented in Terms of a Variable z)

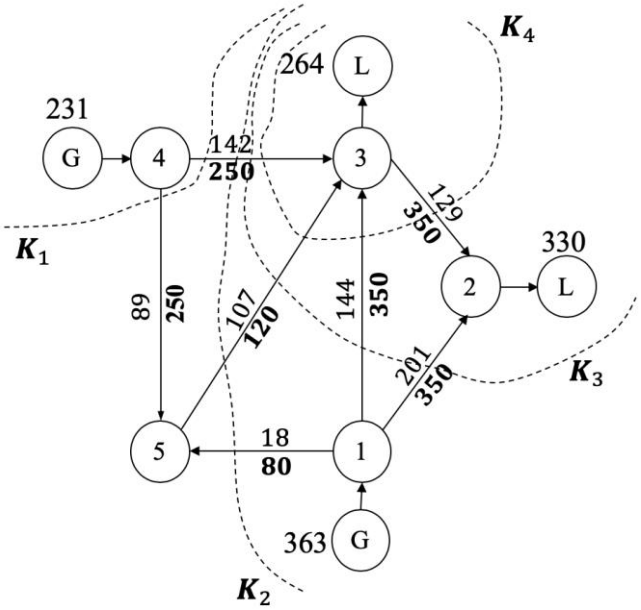

Fig. 3.11 Power Transfer Across Four Different Cut-sets (K_1, K_2,K_3, K_4) Associated with Branch 3-4 for Case 1

Table 3.5 Power Transfer Capacity Across Different Cut-sets in the 5-bus Test System Associated With Branch 3-4

Cut-set	Case 1		Case 2	
	Flow (P_K)	Capacity (R_K)	Flow (P_K)	Capacity (R_K)
K_1	231 MW	250 MW	189 MW	250 MW
K_2	231 MW	200 MW	189 MW	200 MW
K_3	594 MW	820 MW	486 MW	820 MW
K_4	264 MW	820 MW	216 MW	820 MW

Fig. 3.12 presents a DC power flow solution when the total load and total generation of the system is 486 MW (Case 2). In this case, the FT algorithm detects that the indirect paths of branch 3-4 have positive transfer margins indicating that they have the capacity to carry additional power, if need be. To validate this observation, the power transfer capability across each cut-set associated with branch 3-4 is enumerated from the DC power flow solution (see Table 3.5). It is observed that P_K is less than R_K for K_1, K_2, K_3, K_4. This proves that for Case 2, outage of branch 3-4 does not saturate any cut-set that is associated with it.

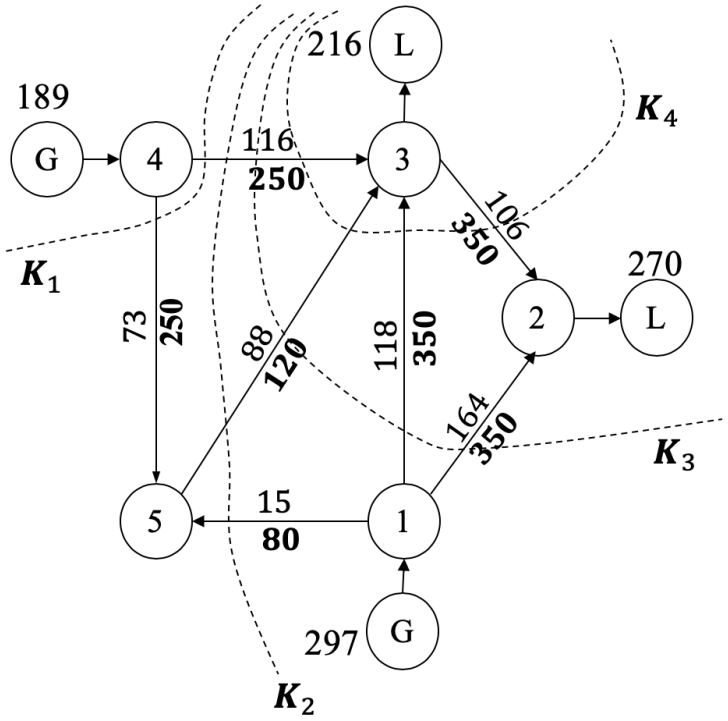

Fig. 3.12 Power transfer Across Four Different Cut-sets (K_1, K_2,K_3, K_4) Associated with Branch 3-4 for Case 2

Furthermore, note that in Fig. 3.11, the power flowing through different branches of the limiting critical cut-set, K_2 ={3-4,3-5,1-5}, are not in the same direction. This implies that cut-set K_2 is not a *coherent* cut-set (in a coherent cut-set power flows in the same direction in all the branches of the cut-set [92]). Therefore, such types of critical

interconnections cannot be detected by the algorithm presented in [92]. It is also important to highlight here that enumerating the power transfer capability across different cut-sets by a DC power flow solution requires previously defining all the cut-sets. On the other hand, the graph theory-based FT can investigate the power transfer capability of different cut-sets without the cut-sets being pre-defined. This is *a unique advantage of the proposed network analysis, because listing all possible cut-sets for a large power network containing thousands of buses especially during extreme event scenarios is not practically feasible*.

CHAPTER 4

MITIGATION OF SATURATED CUT-SETS IN POWER SYSTEMS

This Chapter presents a two-component methodology to enhance the reliability of large power systems during a series of outages. The proposed research is specifically aimed at minimizing the risk of *cascade triggering contingencies* in power systems by enhancing the *N*-1 security after an outage has occurred. The first component demonstrates how the detection and mitigation schemes for alleviating saturated cut-sets can be integrated with the traditional RTCA-SCED framework. As such, this component enhances the scope of existing methods of power system security assessment. The second component proposes an alternative, computationally efficient approach to secure power systems against post-contingency cut-set saturation quickly. The two components are implemented in parallel with the understanding that the solution of the second component will be used only when the more comprehensive first component cannot provide a solution before the next redispatch occurs.

4.1 RTCA and SCED for Real-time Power System Operations

RTCA and SCED are usually employed by power system operators to operate the system in a secure manner [27]-[28]. Fig. 4.1 shows a schematic of state-of-the-art RTCA-SCED framework that takes its inputs from the state estimator. SCED finds a least cost redispatch solution to eliminate the potential post-contingency branch overloads identified by RTCA. The solution obtained by SCED is fed back into the RTCA to ensure that the new solution does not create additional overloads. When no additional violations are detected, the redispatch solution is implemented in the power system.

It was explained in Section 1.2.2 that a subset of the contingencies (selected from operator experience or day ahead studies) are evaluated by RTCA. As the contingency list is *not exhaustive*, it is possible that an important contingency is left out from this list, due to which it is not detected by RTCA (and hence *not* corrected by SCED) until it is too late. This is a serious limitation especially during extreme event scenarios when successive outages occur quickly. Further, when multiple outages have already occurred, a larger number of post-contingency overloads manifest, because the system is in a stressed operating condition. Therefore, SCED takes longer time to find a solution due to the increased number of security constraints that it has to model. Nevertheless, the SCED employs different rounding conventions of PTDFs and approximations in the dispatch model to enhance the computation speed [107]. The increased solution time under extreme scenarios might encourage power system operators to use larger approximations in the model, which would then affect the solution quality. Thus, both the *scope* as well as the *speed* of traditional power system security assessment must be enhanced during multiple outage scenarios.

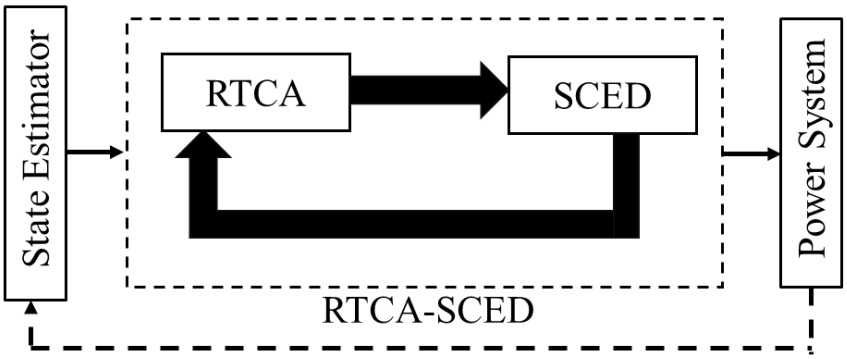

Fig. 4.1: RTCA and SCED for Real-time Power System Operations

4.2 The First Component of the Proposed Methodology

The proposed *first component* aims to make the power system secure against post-contingency cut-set saturation as well as critical branch overloads by integrating the results

from FT and RTCA to create an integrated corrective action (iCA) as shown in Fig. 4.2. The objective of the iCA is to find a least cost re-dispatch solution to ensure that the critical contingencies detected by RTCA do not create post-contingency branch overloads and the special assets identified by FT do not create saturated cut-sets. During multiple outage scenarios, it is possible that a re-dispatch solution is not able to mitigate all the identified overloads. Under such circumstances, controlled load shedding will be implemented. Since disconnecting the loads incur high economic and social costs [108], load-shedding will be used as the last resort during redispatch.

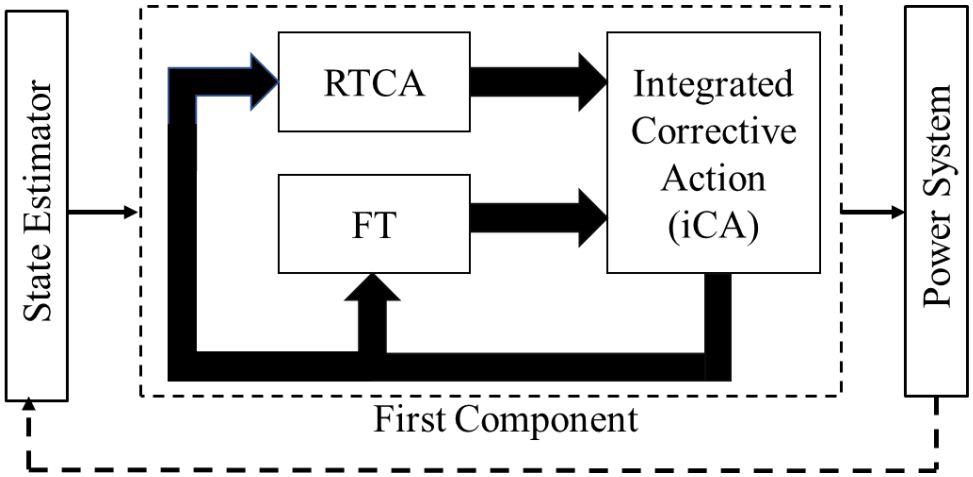

Fig. 4.2 The First Component: The Results from RTCA and FT are Used to Create an Integrated Corrective Action (iCA)

Consider that the generator at bus $i \in G$ in the system is associated with a quadratic cost curve as shown below:

$$F_i(G_i) = a_i + b_i G_i + c_i G_i^2 \tag{4.1}$$

where, G_i is the power produced (in MW) by the generator at bus i, and a_i, b_i, and c_i are the fixed cost coefficient (in \$), the linear cost coefficient (in \$/MW), and the quadratic cost coefficient (in \$/MW^2), respectively, for the corresponding generator. Let G_i^o and G_i^n

denote the power produced before and after the new dispatch. The change in generation cost as a function of change in power generation, $\Delta G_i (= G_i^n - G_i^o)$, is given by,

$$\begin{aligned}\Delta F_i(\Delta G_i) &= \{a_i + b_i G_i^n + c_i (G_i^n)^2\} - \{a_i + b_i G_i^o + c_i (G_i^o)^2\} \\ &= b_i (G_i^n - G_i^o) + c_i \{(G_i^n)^2 - (G_i^o)^2\} \\ &= b_i (G_i^n - G_i^o) + c_i (G_i^n + G_i^o)(G_i^n - G_i^o) \\ &= b_i (\Delta G_i) + c_i (2G_i^o + \Delta G_i)(\Delta G_i) \\ &= b_i (\Delta G_i) + (2c_i G_i^o + c_i \Delta G_i)\Delta G_i \\ &= c_i \Delta G_i^2 + (b_i + 2c_i G_i^0)\Delta G_i \end{aligned} \quad (4.2)$$

where, $d_i = (2c_i G_i^o + b_i)$. Now, the cost of shedding the load at bus $j \in L$ can be written as follows:

$$\Delta F_j(\Delta L_j) = m_j \Delta L_j \quad (4.3)$$

where, ΔL_j denotes the amount of load-shed, and m_j is the cost coefficient of load-shed (in \$/MW); m_j is chosen to be significantly higher compared to the generator cost coefficients, because the goal is to use load-shed only when generation redispatch alone cannot mitigate all violations. The convex optimization problem that minimizes the total cost of change in generation and load-shed is given by:

$$\text{Minimize: } \sum_{\forall i \in G} (c_i \Delta G_i^2 + d_i \Delta G_i) + \sum_{\forall j \in L} (m_j \Delta L_j) \quad (4.4)$$

The constraints to be applied to (4.4) are as follows.

4.2.1 Branch Power Flows

To model the branch power flow limits PTDFs are used. It has been explained in Section 1.2.3 that PTDFs are linear sensitivity factors that approximate the change in flow

through a branch caused by a change in power injection in the system. Let $PTDF_{l,i}^{r}$ denotes the change in flow in branch e_l, for one unit of power added at bus i and one unit of power withdrawn from the reference bus of the system. Then, the change in flow, Δf_l, through e_l for the change in bus power injections can be obtained as follows:

$$\Delta f_l = \sum_{\forall i \in G} PTDF_{l,i}^{r} \Delta G_i - \sum_{\forall j \in L} PTDF_{l,j}^{r} \Delta L_j \tag{4.5}$$

Consequently, the constraint equation for the maximum and minimum power flows is given as follows:

$$\sum_{\forall i \in G} PTDF_{l,i}^{r} \Delta G_i - \sum_{\forall j \in L} PTDF_{l,j}^{r} \Delta L_j \leq f_l^{max} - f_l^{0}, \qquad \forall e_l \in E \tag{4.6}$$

$$\sum_{\forall i \in G} PTDF_{l,i}^{r} \Delta G_i - \sum_{\forall j \in L} PTDF_{l,j}^{r} \Delta L_j \geq f_l^{min} - f_l^{0}, \forall e_l \in E \tag{4.7}$$

where, f_l^{o}, f_l^{max} and f_l^{min} denote the original power flow, maximum power flow limit, and the minimum power flow limits, respectively.

4.2.2 Power Injections

The maximum and minimum power production constraints for the generators are given as follows:

$$\Delta G_i \leq G_i^{max} - G_i^{0}, \quad \forall i \in G \tag{4.8}$$

$$\Delta G_i \geq G_i^{min} - G_i^{0}, \quad \forall i \in G \tag{4.9}$$

where, G_i^{o}, G_i^{max}, and G_i^{min} denote the original power production, maximum power production and minimum power production of the generator at bus i, respectively. Similarly,

the constraints for the minimum and maximum power demand at a load bus j are given as follows:

$$\Delta L_j \leq L_j^{max} - L_j^0, \qquad \forall j \in L \tag{4.10}$$

$$\Delta L_j \geq L_j^{min} - L_j^0, \qquad \forall j \in L \tag{4.11}$$

4.2.3 Conservation of Energy

To ensure the conservation of energy, the aggregate change in generation dispatch must equal the net change in power demand in the system.

$$\sum_{\forall i \in G} \Delta G_i = \sum_{\forall j \in L} \Delta L_j \tag{4.12}$$

4.2.4 Security Constraints 1: Post-contingency Branch Flows

The post-contingency branch flow constraints can be efficiently modeled with the LODFs [109]. Consider that $\text{LODF}_{l,k}$ represents the percentage of change in flow through branch e_k that will appear on branch e_l for an outage of branch e_k (refer to Section 1.2.3). The post-contingency flow through e_l for a potential outage of branch e_k is given as:

$$f_l^c = f_l^n + LODF_{l,k} f_k^n \tag{4.13}$$

where, f_l^n and f_k^n denote the new flows corresponding to the iCA solution through branches e_l and e_k respectively. Equation (4.13) could be re-written as follows:

$$f_l^c = (f_l^0 + \Delta f_l) + LODF_{l,k}(f_k^0 + \Delta f_k) \tag{4.14}$$

where, f_l^0 and f_k^0 denote the original flows through branches e_l and e_k respectively. Similarly, Δf_l and Δf_k represent the incremental change in branch-flows e_l and e_k as obtained from the redispatch. Substituting Δf_l and Δf_k from (4.5) into (4.14), and using the

respective branch flow limits, we obtain the equations for the post-contingency branch flow constraints:

$$\left\{\begin{matrix}\sum_{\forall i\in G}\left(PTDF_{l,i}^{r}+\ LODF_{l,k}PTDF_{k,i}^{r}\right)\Delta G_i- \\ \sum_{\forall j\in L}\left(PTDF_{l,j}^{r}+\ LODF_{l,k}PTDF_{k,j}^{r}\right)\Delta L_j\end{matrix}\right\}\leq f_l^{max}-\left(f_l^0+LODF_{l,k}f_k^0\right)$$

$$\forall e_k\in E_v\ ,\forall e_l\in E \quad (4.15)$$

$$\left\{\begin{matrix}\sum_{\forall i\in G}\left(PTDF_{l,i}^{r}+\ LODF_{l,k}PTDF_{k,i}^{r}\right)\Delta G_i- \\ \sum_{\forall j\in L}\left(PTDF_{l,j}^{r}+\ LODF_{l,k}PTDF_{k,j}^{r}\right)\Delta L_j\end{matrix}\right\}\geq f_l^{min}-\left(f_l^0+LODF_{l,k}f_k^0\right)$$

$$\forall e_k\in E_v\ ,\forall e_l\in E \quad (4.16)$$

where, set E_v contains the critical contingencies detected by RTCA. The constraints (4.15) and (4.16) are modeled for all post-contingency branch overloads for the critical contingencies detected by RTCA [109].

4.2.5 Security Constraints 2: Cut-set Power Transfer

This type of security constraints is designed for the special assets detected by the FT algorithm. The objective here is to reduce the total power transfer across the limiting critical cut-set K_{crit} by the respective transfer margin T_l as follows:

$$\sum_{\forall e_l\in K_{crit}}\Delta f_l\leq T_l, \quad (4.17)$$

where, Δf_l denotes the change in flow through branch e_l. Now, substituting Δf_l from (4.5) to (4.17), the constraints for cut-set power transfer are obtained as follows:

$$\sum_{\forall i\in G}\left(\sum_{\forall e_l\in K_{crit}}PTDF_{l,i}\right)\Delta G_i-\sum_{\forall j\in L}\left(\sum_{\forall e_l\in K_{crit}}PTDF_{l,j}\right)\Delta L_j\leq T_l$$

$$\forall\ K_{crit}\in\mathcal{K}_{crit} \quad (4.18)$$

where, the set $\mathcal{K}_{crit}$ contains the limiting critical cut-sets detected by the FT corresponding to different special assets.

Note that a SCED essentially solves the same optimization problem as the iCA with all constraints modeled except the cut-set power transfer constraints [64]. By considering both post-contingency branch overloads as well as post-contingency cut-set saturation, the iCA creates a more comprehensive corrective action than the SCED.

4.3 The Second Component of the Proposed Methodology

The first component of Section 4.2 (or the traditional RTCA-SCED framework of Section 4.1) are likely to take more time because of the larger number of security constraints modeled in the optimization problem for iCA (or SCED). For example, if the number of critical contingencies detected by RTCA is $|E_v|$, and the total number of transmission assets is $|E|$, the number of post-contingency branch flow constraints (see *security constraints 1* in Section 4.2.4) that must be modeled is $|E_v| \times |E|$. For a large power system, containing thousands of branches, $|E|$ is large. Moreover, for a stressed power system that has suffered multiple outages, $|E_v|$ is also large. Consequently, the proposed first component (or the RTCA-SCED) will not be able to suggest corrective actions at high speeds.

To provide a high-speed corrective action, a second component is proposed, which only utilizes the results from FT to create a relaxed corrective action (rCA) as shown in Fig. 4.3. The rCA solves the same optimization problem (given by (4.4)), but *without modeling the post-contingency branch flow constraints* (described by (4.15) and (4.16)). However, the cut-set power transfer constraints, described by (4.18), are retained in rCA, i.e., the rCA utilizes the results from FT to only secure the system against post-contingency

cut-set saturation. Note that if the optimization problem given by (4.4) is solved without modeling any security constraints (neither security constraints 1, nor security constraints 2), it reduces to a simple DC optimal power flow (DC-OPF) problem. Therefore, by considering the cut-set power transfer constraints (security constraints 2), the rCA adds a relaxed criterion of power system security onto an OPF problem.

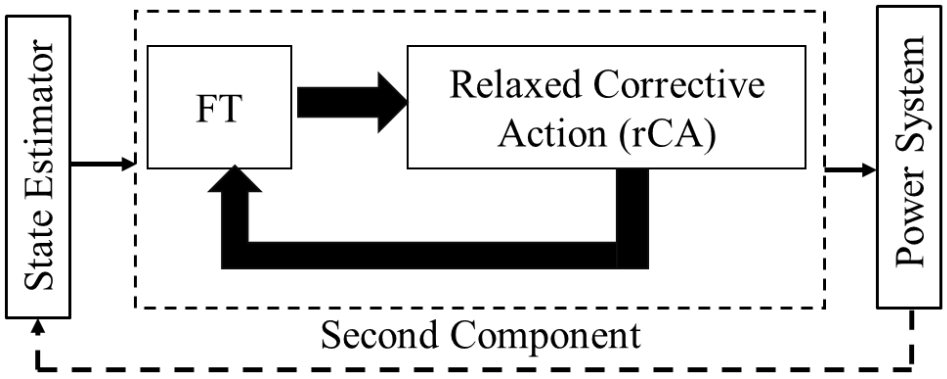

Fig. 4.3 The Second Component: The Results from FT are Only Utilized to Create a Relaxed Corrective Action (rCA)

If the set E_s contains the special assets detected by FT, the number of cut-set power transfer equations modeled by the rCA is $|E_s|$. Now, as the number of cut-set violations identified will be smaller than the total number of branches of a power system, $|E_s| \ll |E|$, and consequently, $|E_s| \ll |E_v| \times |E|$. *This implies that the number of security constraints modeled by the rCA is significantly less compared to the number of security constraints modeled by the iCA (or SCED)* and is the primary reason for the very high speed of rCA.

It should however be noted that the solution obtained using the second component is secure against pre-contingency branch overloads and post-contingency cut-set saturation, but not post-contingency branch overloads. Conversely, the solution obtained from the first component is secure against post-contingency cut-set saturation, as well as pre-contingency and post-contingency branch overloads. Naturally, the solution quality of the first component is better than the second.

At the same time, it is important to note that if generation redispatch alone cannot provide a feasible solution with respect to a relaxed set of constraints such as those used in rCA, it is obvious that generation redispatch will not provide a solution with more comprehensive constraints such as those used in iCA. Therefore, if load-shedding is indicated by rCA (in the second component), it will also be indicated by iCA (in the first component); albeit after a longer time and the amount of load-shed will be equal or higher. Therefore, the ability to quickly indicate the minimum amount of load that must be shed before a detailed network analysis tool can provide a more accurate estimate of load-shed, is another advantage of the rCA.

4.4 Real-time Application of the Proposed Two-component Methodology

It can be realized from Sections 4.2 and 4.3 that the first and second components enhance the scope and speed, respectively, of traditional power system security assessment. The question then becomes, *how should the two components be applied in real-time when a contingency occurs?* Different entities implement SCED at different timescales for real-time power system operations. For example, PJM Interconnection LLC implements real-time SCED every fifteen minutes [110], whereas Midcontinent Independent System Operator (MISO) implements SCED every five minutes [111]. In this context, the real-time application of the two components can be explained using timelines shown in Fig. 4.4.

With reference to Fig. 4.4, let an outage occur at time t_o. Following the outage, the first and second components should be initiated simultaneously but independently. Let the redispatch solution be implemented at time t_d, while the first and second components provide their dispatch solutions at time t_i and t_r, respectively. If $t_i < t_d$, as shown in Fig.

4.4(a), then the solution obtained using the first component should be used for redispatch as it has better quality. However, if $t_i > t_d$ and $t_r < t_d$, as shown in Fig. 4.4(b) then the solution obtained from the second component should be implemented to at least secure the system against post-contingency cut-set saturation. It will be shown in Section 5.2.2 that the computational burden of the second component is comparable to a simple DC-OPF. As such, the likelihood of $t_r > t_d$ is small even for large power systems. However, if that still happens then depending on its availability, the solution from the first (preferred) or the second component should be implemented in the next redispatch.

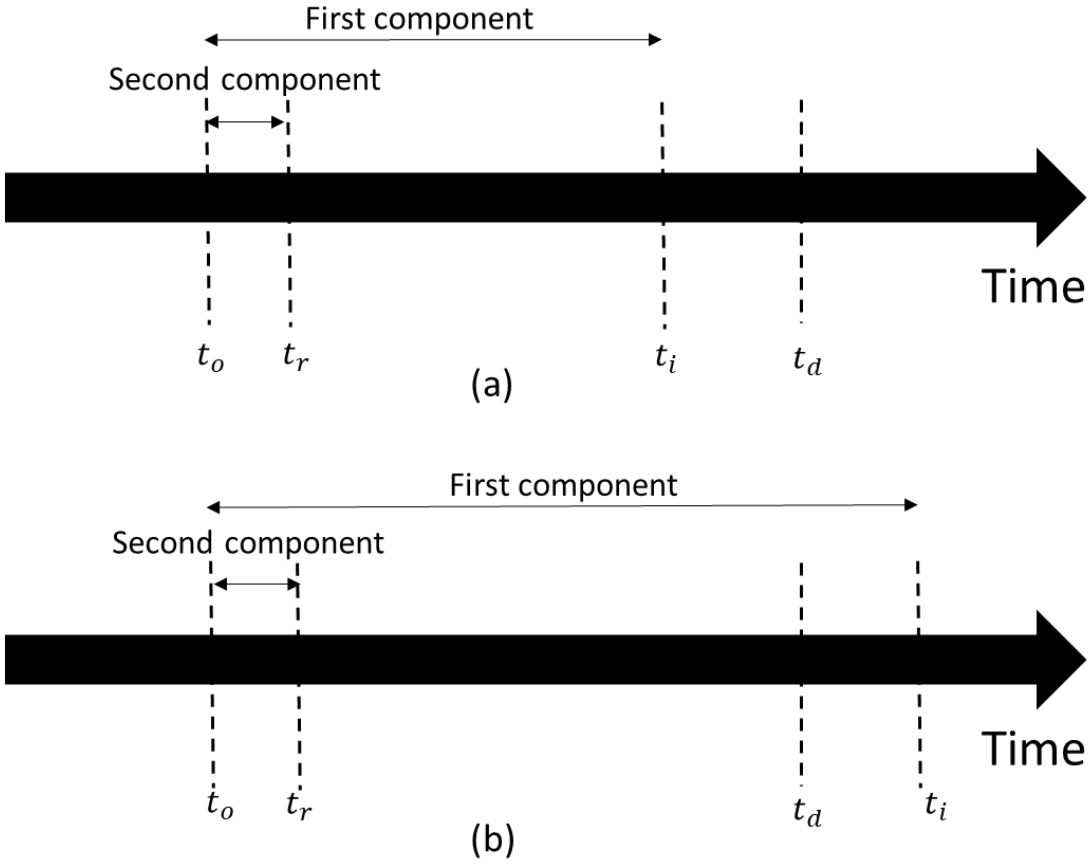

Fig. 4.4 (a) If the First Component Provides a Dispatch Solution Before the Scheduled Time for the Next Redispatch, then the Solution Obtained from the First Component Should be Implemented, (b) If the First Component Does not Provide a Dispatch Solution Before the Scheduled Time for Next Redispatch, then the Solution Obtained from the Second Component Should be Implemented

4.5 The Modified Update Scheme (M-UPS) Algorithm

The corrective actions introduced by iCA (in the first component) and rCA (in the second component) change the bus power injections. Therefore, FT must re-evaluate the system corresponding to the new bus power injections to ensure that the updated system does not have any additional saturated cut-sets due to a potential outage. Hence, a modified-update scheme (M-UPS) is developed in this dissertation that updates the flow and latent capacity graphs in a computationally efficient manner, thereby eliminating the need for recreating these weighted graphs from scratch. Let the sets V^p and V^n contain the buses where the power injection has increased and decreased, respectively. Increase in the net power injection at a bus refers to either generation being increased, or load being decreased. Similarly, decrease in net power injection at a bus refers to either generation being decreased, or load being increased. Let, ΔI_p and ΔI_n denote the increase and decrease in net power injection at buses $v_p \in V^p$ and $v_n \in V^n$, respectively. Now the updated flow and latent capacity graphs can be obtained using Algorithm IV.

4.5.1 Illustration of the M-UPS algorithm

The flow and the latent capacity graph of a sample 5-bus test system is shown in Fig. 4.5. Let us consider that the corrective action (either iCA in the first component or rCA in the second component) reduces the generation at bus 4 by 30 MW and reduces load at bus 2 by 30 MW. The iterations of the M-UPS algorithm are explained as follows. The sets V_p and V_n of Algorithm IV are given as follows: $V_p = \{2\}$ $V_n = \{4\}$.

Algorithm IV: Modified Update Scheme (M-UPS)

i. Randomly select a source $v_p \in V^p$ and a sink $v_n \in V^n$.

ii. Search $\mathcal{C}(V, E)$ to traverse the shortest unsaturated path $\mathcal{P}$ from v_p to v_n using breadth first search (BFS) [95].

iii. Use $\mathcal{C}$ to find the maximum extra flow, $C_{\mathcal{P}}$, that can be transferred from v_p to v_n through path $\mathcal{P}$.

iv. Obtain the flow, $F_{\mathcal{P}}$, to be injected in $\mathcal{F}(V, E)$ along path $\mathcal{P}$ from v_p to v_n as $F_{\mathcal{P}} = \min(\Delta I^p, \Delta I^n, C_{\mathcal{P}})$.

v. Update weights of branches in graph $\mathcal{F}$ as $f_l = f_l + F_{\mathcal{P}}$, and in graph $\mathcal{C}$ as $c_l^{FT} = c_l^{FT} - F_{\mathcal{P}}$ and $c_l^{TF} = c_l^{TF} + F_{\mathcal{P}}$, for all branches that belong to path $\mathcal{P}$.

vi. Update net power injections at v_p and v_n as $\Delta I^p \coloneqq \Delta I^p - F_{\mathcal{P}}$ and $\Delta I^n \coloneqq \Delta I^n - F_{\mathcal{P}}$.

vii. Depending upon the values of ΔI^p and ΔI^n, update the source and sink in accordance with the following logic:

   a. if $\Delta I^p \neq 0$ & $\Delta I^n \neq 0$, the source and sink are not changed.

   b. if $\Delta I^p = 0$ & $\Delta I^n \neq 0$, a new source v_p is selected from set V^p, keeping the sink v_n, unchanged.

   c. if $\Delta I^p \neq 0$ & $\Delta I^n = 0$, a new sink is selected from set V_n, keeping the source v_p, unchanged.

viii. Repeat Steps (ii) through (vii) until the total increase in power injection is compensated by the total decrease in power injection.

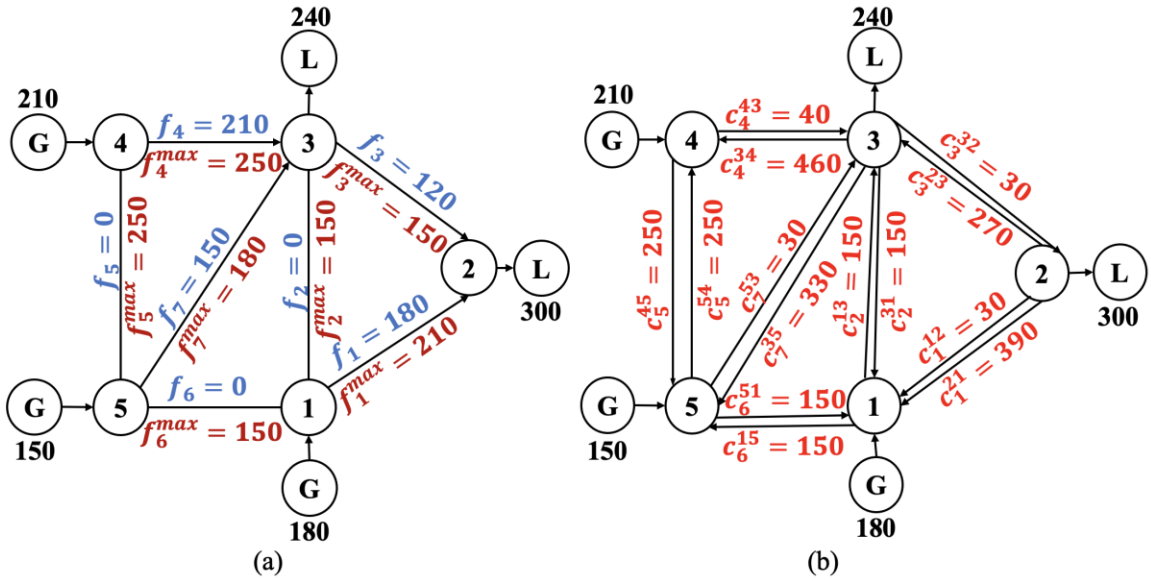

Fig. 4.5 (a) Flow Graph and (b) Latent Capacity Graph for a sample 5-bus test system

Iteration 1:

Step i: A source v_p and a sink v_n are to be selected from sets V_p and V_n, respectively; so, $v_p = 2$ and $v_n = 4$. ΔI_p and ΔI_n are 30 MW each.

Step ii: The shortest unsaturated path from bus 2 to bus 4 is given as follows: $\mathcal{P} = \{2 - 3 - 4\}$.

Step iii: The maximum power that could be re-routed from bus 2 to bus 4 is 270 MW; $C_{\mathcal{P}}$ =270. This can be observed from the latent capacity graph of Fig. 4.5(b). We observe that along path $2 - 3 - 4$, branch 2-3 is limiting because it has a lower latent capacity of 270 MW.

Step iv: The flow $F_{\mathcal{P}}$ that must be injected in the flow graph along path $\mathcal{P}$ from bus 2 to bus 4 is as follows:

$$F_{\mathcal{P}} = \min\left(\Delta I_p, \Delta I_n, C_{\mathcal{P}}\right) = \min(30,30,270) = 30 \tag{4.19}$$

Step v: The weights of the branches in the flow and latent capacity graphs are updated, for an injection of 30 MW of flow along path $\mathcal{P}$ (see Fig. 4.6).

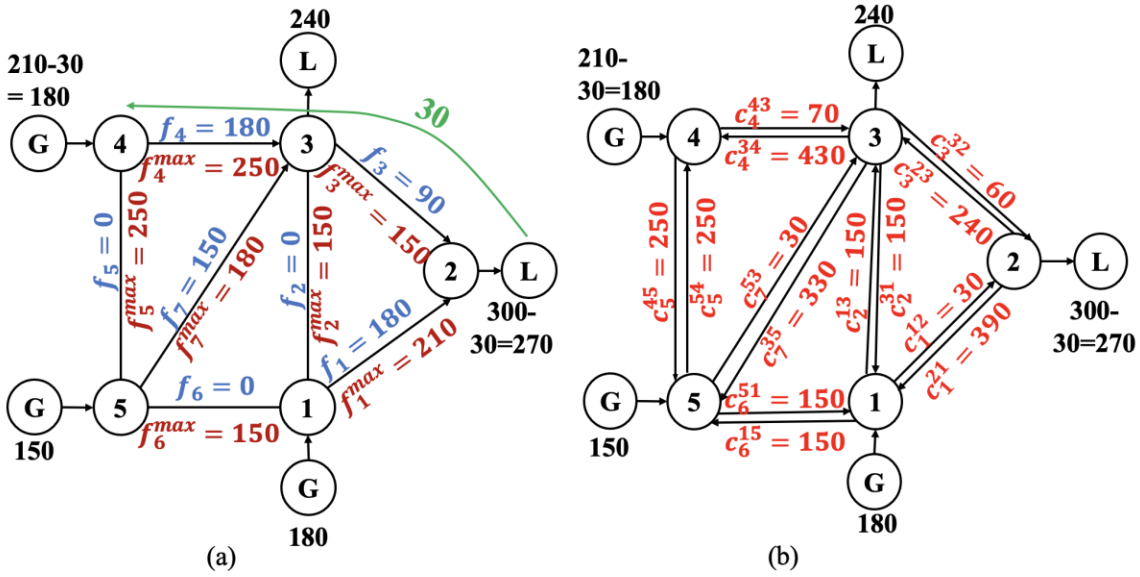

Fig. 4.6 (a) Updated Flow Graph, and (b) Latent Capacity Graph Obtained After a Redispatch Solution

Step vi: ΔI_p and ΔI_n are updated as follows:

$$\Delta I_p = \Delta I_p - C_{\mathcal{P}} = 30 - 30 = 0 \tag{4.20}$$

$$\Delta I_n = \Delta I_n - C_{\mathcal{P}} = 30 - 30 = 0 \tag{4.21}$$

Step v: Since, $\Delta I_p = 0$ and $\Delta I_n = 0$ and there are no additional buses in sets V^p and V^n, the M-UPS algorithm is terminated.

We observe from the above example that the M-UPS algorithm creates an updated flow graph utilizing the set of shortest indirect paths to re-route the flows. *This is possible because in the context of detecting saturated cut-sets, the net power transfer across any cut-set of the network is important, rather than the individual branch flows.* Since it does not matter which paths are selected to match the total load with generation, following a system redispatch, the set of shortest indirect paths can be used to re-route the flows using Algorithm IV. This is explained with the help of another flow solution obtained from DC power flow after generation dispatch.

Fig. 4.7(a) and 4.7(b) present the flow and latent capacity graphs obtained from a DC power flow. Fig. 4.8(a) and 4.8(b) compares the flow graphs obtained from the graph-theory based M-UPS algorithm and the DC power flow solution respectively. Despite the individual branch flows being different, the power transfer across any cut-set of the network remains constant. For example, the total power transfer across cut-set K_1 is 330 MW in both the graphs. Consequently, if the FT is applied on any flow solution, it detects that the outage of branch 4-3 will not saturate cut-set K_1 beyond its capacity (transfer margin equals zero). This is because the total power transfer capacity of cut-set K_1 reduces to exactly 330 MW after the outage of branch 4-3.

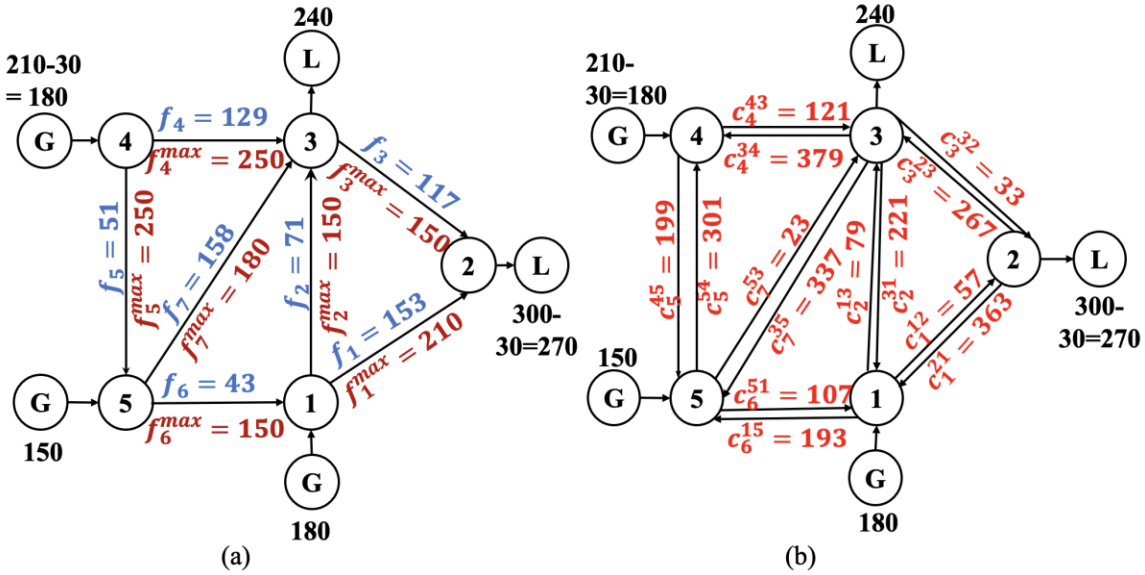

Fig. 4.7: (a) Flow Graph, and (b) Latent Capacity Graph Obtained from a DC Power Flow Solution After Generation Redispatch

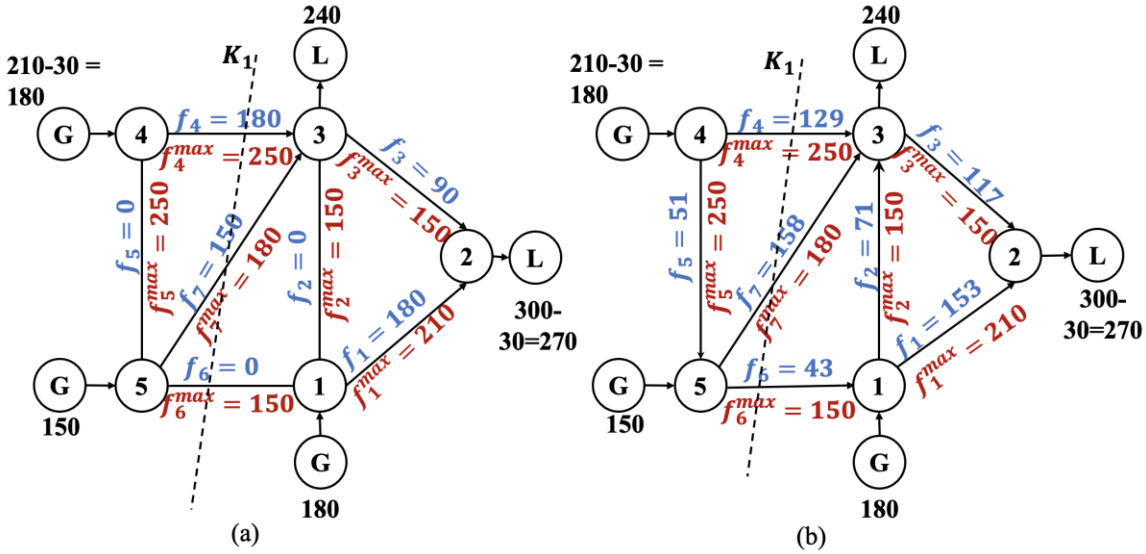

Fig. 4.8 (a) Flow Graph Obtained from the M-UPS Algorithm, and (b) Flow graph obtained from a DC Power Flow Solution After Generation Redispatch

4.6 The Modified Shortlisting Assets (M-SA) Algorithm

In the pre-outage scenario, all assets are evaluated by the FT. However, once the M-UPS creates an updated flow graph it may not be necessary to evaluate all assets by FT once again to identify the set of special assets. Hence, a modified-shortlisting asset (M-SA) scheme is developed in this dissertation which finds the contingencies to be evaluated by FT following the update of the flow graph to account for the changes in bus power injections.

The concept of M-SA is explained with the help of Fig. 4.9. Let the M-UPS modify the flows through path $\mathcal{P}_2$ in the network to account for the changes in bus power injections. Also, from the FT performed in the pre-outage scenario, let it be known that the flow of another branch e_m can be re-routed through path $\mathcal{P}_1$. Now, if paths $\mathcal{P}_1$ and $\mathcal{P}_2$ do not have any common branches as shown in Fig. 4.9(a); FT need not be repeated for branch e_m. This is because we already know from the pre-outage scenario analysis that the outage of e_m does not saturate a cut-set and the disrupted flow can be re-routed through path $\mathcal{P}_1$ itself.

However, if paths $\mathcal{P}_1$ and $\mathcal{P}_2$ have branches in common as shown in Fig. 4.9(b), then e_m must be re-evaluated by the FT, once the network flows have been updated.

It must be noted here that the proposed M-UPS and M-SA algorithms are used to perform a successive FT, when the corrective actions made by the iCA and rCA change the bus power injections. Conversely, the original UPS and SA proposed in Sections 2.6 and 2.7 were used to perform a successive FT following a branch outage that have occurred in the system.

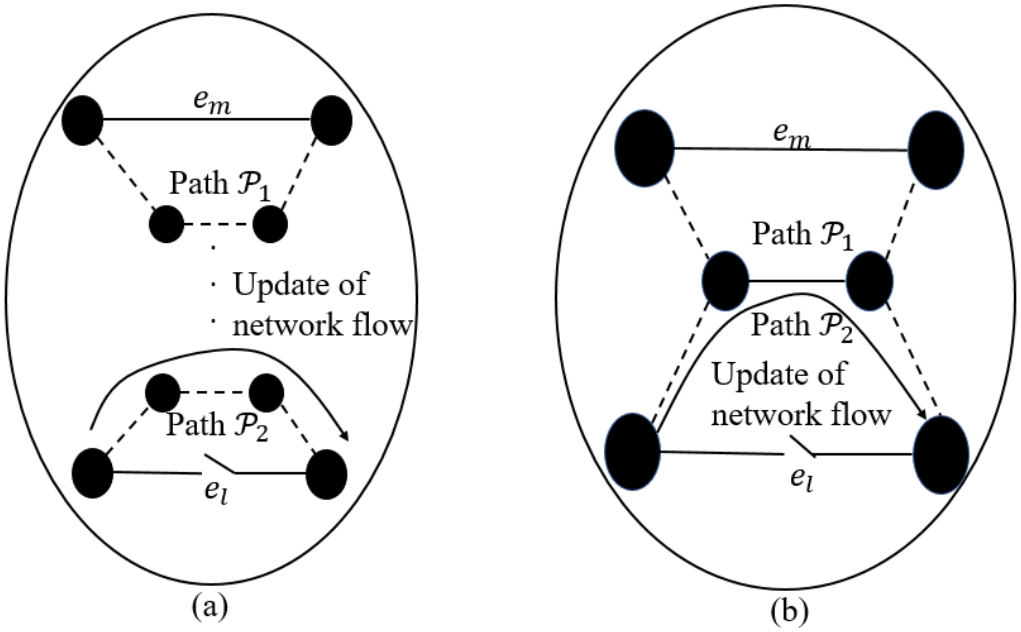

Fig. 4.9 (a) Updating the Flows in the Network for a Change in the Power Injections Does not Involve any Branch in the Indirect Paths of Branch e_m; (b) Updating the Flows in the Network for a Change in the Power Injections Involves Branches in the Indirect Paths of Branch e_m

4.6.1 Illustration of the Modified Shortlisting Assets (M-SA) algorithm

Fig. 4.10 shows the original flow graph of a sample 7-bus power system. Let the corrective action schemes (either iCA or rCA) sheds 20 MW of load at bus 1 and reduces 20 MW of generation at bus 3. To account for the changes in bus power injections the M-UPS reroutes 20 MW of flow from bus 1 towards bus 3 along path along $\mathcal{P} = \{1 - 2 - 3\}$.

As such, the flows through branches 1-2 and 2-3 are updated to create a new flow graph, as shown in Fig. 4.11. These branches have been highlighted in green to indicate that the graph-theory based network flows have been updated.

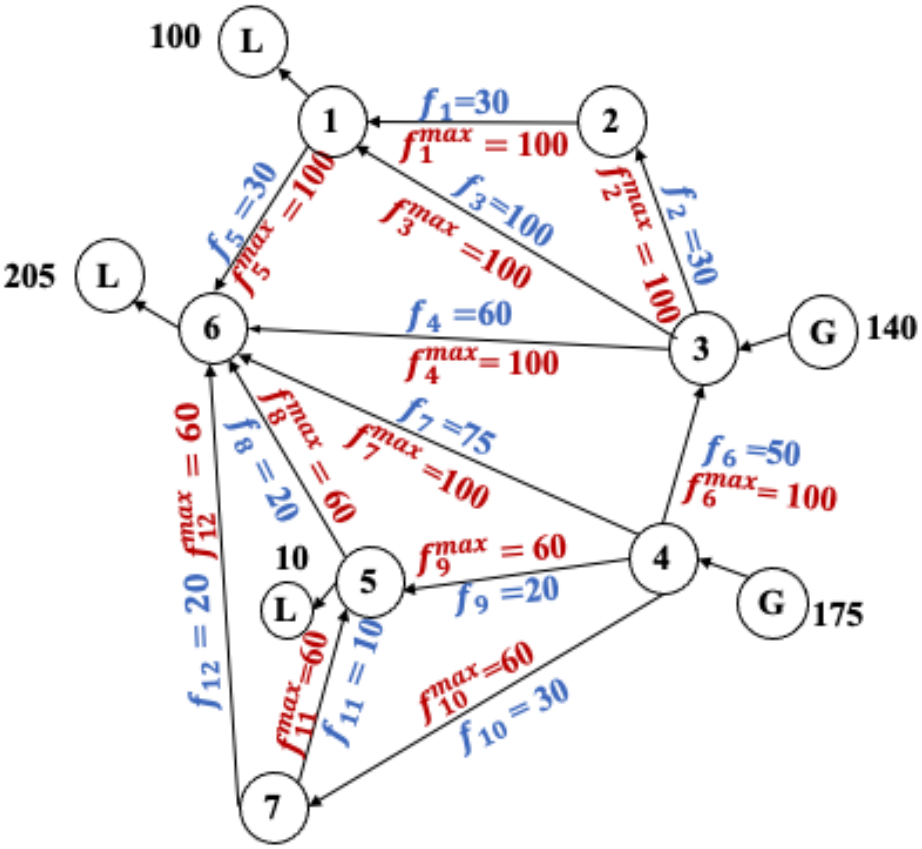

Fig. 4.10: The Flow Graph of a Sample 7-bus Power System Before the Corrective Action has been Implemented

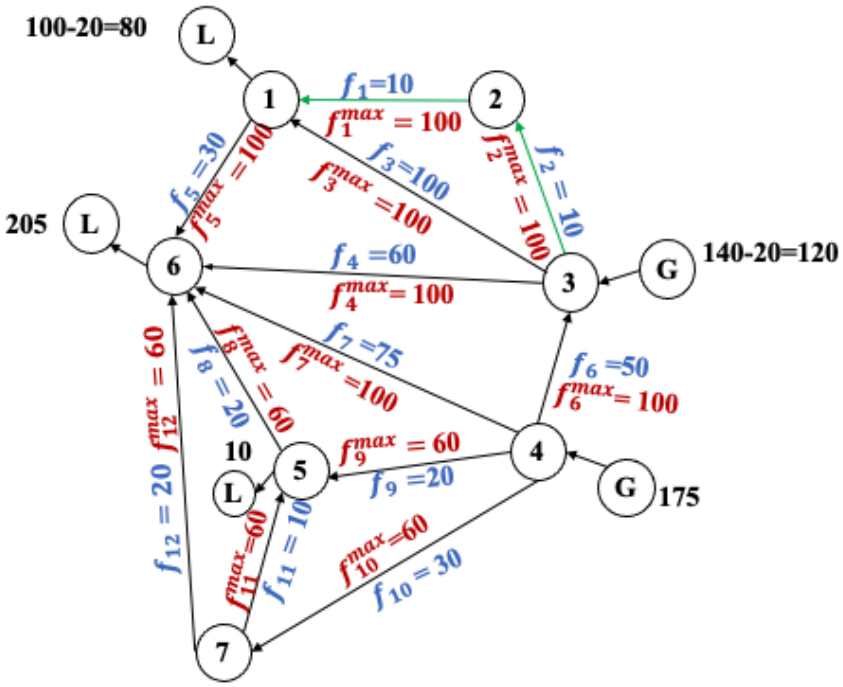

Fig. 4.11 The Flow Graph of a Sample 7-bus Power System After the Corrective Action has been Implemented

Table 4.1 enumerates the information that was obtained from the FT in the pre-correction scenario (before the corrective actions have been initiated). For example, the third row of Table 4.1 implies that for the outage of branch 3-1, the flows can be re-routed along paths 3-6-1, 3-4-6-1, and 3-4-5-6-1. The specific transmission assets whose *indirect paths* involve the branches 1-2 and 2-3 (for which the flows are updated), must be re-evaluated by the FT, after the new dispatch solution is applied on the graphs. With this rationale only branches 2-1 and 3-2 must be re-evaluated by the FT in this example, after the M-UPS has updated the graphs. The other branches need not be re-evaluated by the FT.

Table 4.1 Information of the FT Before the New Dispatch Solution is Obtained

Feasibility Test (FT) for an edge	Indirect paths through which rerouting of flow can occur
Branch 1-6	Indirect Path 1: 1-3-6
Branch 2-1	Indirect Path 1: 2-3-6-1
Branch 3-2	Indirect Path 1: 3-6-1-2
Branch 3-1	Indirect Path 1: 3-6-1 Indirect Path 2: 3-4-6-1 Indirect Path 3: 3-4-5-6-1
Branch 3-6	Indirect Path 1: 3-4-6 Indirect Path 2: 3-4-5-6
Branch 4-3	Indirect Path 1: 4-6-3 Indirect Path 2: 4-5-6-3
Branch 4-6	Indirect Path 1: 4-3-6 Indirect Path 2: 4-5-6
Branch 5-6	Indirect Path 1: 5-4-6
Branch 4-5	Indirect Path 1: 4-6-5
Branch 4-7	Indirect Path 1: 4-5-7
Branch 7-6	Indirect Path 1: 7-5-6
Branch 7-5	Indirect Path 1: 7-4-5

CHAPTER 5

RESULTS: MITIGATION OF SATURATED CUT-SETS

This Chapter evaluates performance of the two-component methodology for mitigation of post-contingency cut-set saturation on the IEEE 118-bus test system and the 2000-bus synthetic Texas system. A comparative study of the proposed methodology with traditional approaches, such as RTCA-SCED and DC-OPF, is also presented. All simulations were performed in MATLAB. GUROBI was used to solve the optimization problems.

5.1 Mitigation of Saturated Cut-sets in the IEEE 118-bus Test System

We initially present the performance of the proposed two-component methodology using a detailed case-study that involves a sequence of six outages. Subsequently, to demonstrate consistency, its performance is compared with the traditional approaches for 40 additional case-studies.

5.1.1 A Detailed Case-study of the IEEE 118-bus Test System

The performance of the first component is presented and compared with the RTCA-SCED framework when six outages manifest successively. The first column of Table 5.1 shows the sequence of events. Columns two through six present the results associated with the first component. The second column presents the special assets detected by the FT algorithm. An outage of any of these special assets (after the outage that has already occurred in the corresponding row of the first column), will create post-contingency cut-set saturation. The third column shows the critical contingencies detected by RTCA that result in post-contingency branch overloads. To determine the entries of this column, a two-step

procedure was followed: (a) PTDFs and asset ratings were used to rank the contingencies following every outage [53], and (b) top 30% of the contingencies [27] were evaluated by RTCA to determine the post-contingency branch overloads. The special assets detected by FT in the second column and the critical branch contingencies detected by RTCA in the third column were set as inputs to the iCA.

Next, an independent cascading simulation analysis was conducted using MATCASC [76]. To screen out the contingencies that will trigger a cascade and result in unserved power demand, every outage was evaluated by MATCASC. The fourth and sixth columns of Table 5.1 present the cascade triggering contingencies detected by MATCASC *before* and *after* the implementation of iCA. The fifth column presents the redispatch solution (generation cost) obtained from the iCA. Note that the redispatch solution for this case-study did not result in any load-shed. Finally, we observe from the sixth column that the solution obtained from iCA does not contain any cascade triggering contingencies. *Therefore, through iCA, the first component has effectively utilized the information from FT and RTCA to mitigate cascade triggering contingencies for the given sequence of events.*

Now, we evaluate the performance of the RTCA-SCED framework for the same sequence of events. Columns seven through ten of Table 5.1 present the results associated with RTCA-SCED. The column headings are similar to that of the first component, with the exception that the FT results are absent in this section as the traditional SCED only utilizes the inputs from RTCA. For the first five outages the results of the first component and RTCA-SCED are identical. This is because for the first five outages the FT does not identify additional violations to those already detected by RTCA (compare the second and

third columns of Table 5.1). However, after the sixth outage FT detects the special asset 65-66 in addition to the critical contingency 64-65 identified by RTCA (see second and third column of the last row). This is the basis for the difference in the redispatch solutions of the first component and RTCA-SCED as seen in the fifth and ninth columns of the last row. Finally, it is observed that the RTCA-SCED solution contains one cascade triggering contingency (65-66), while the solution obtained from iCA did not have any (see sixth and tenth columns of the last row). *This observation proves that integrating the results from FT with RTCA enhances the ability of power system security assessment in mitigating the risk of cascade triggering contingencies*.

Table 5.1 Comparative Analysis of the First Component and RTCA-SCED for a Sequence of Outages in the IEEE 118-bus Test System

Event (branch outages)	First component (FT-RTCA-iCA)					RTCA-SCED			
	FT	RT CA	MATC ASC (before correc-tion)	Gen. cost (k$)	MAT CASC (after correc-tion)	RT CA	MATC ASC (before correc-tion)	Gen. cost (k$)	MAT CASC (after correc-tion)
Outage 1: 15-33	-	-	-	126.2	-	-	-	126.2	-
Outage 2: 19-34	-	5-8	-	126.3	-	5-8		126.3	-
Outage 3: 37-38	42-49	42-49, 5-8, 26-30	42-49	126.5	-	42-49 5-8, 26-30	42-49	126.5	-
Outage 4: 42-49	45-46, 45-49	45-46, 45-49	45-46, 45-49	126.7	-	45-46, 45-49	45-46, 45-49	126.7	-
Outage 5: 49-66	-	5-8	-	126.7	-	5-8	-	126.7	-
Outage 6: 66-67	64-65, 65-66	64-65	64-65, 65-66	127.1	-	64-65	64-65, 65-66	126.9	65-66

Table 5.2: Comparative Analysis of the Second Component and DC-OPF for a Sequence of Outages in the IEEE 118-bus Test System

Event (branch outages)	Second component (FT-rCA)				DC-OPF	
	FT	MATCASC (before correction)	Gen. Cost (k$)	MATCASC (after correction)	Gen. Cost (k$)	MATCASC
Outage 1: 15-33	-	-	126.2	-	125.9	26-30
Outage 2: 19-34	-	-	126.2	-	125.9	26-30
Outage 3: 37-38	42-49	42-49	126.3	-	125.9	26-30, 42-49
Outage 4: 42-49	45-46, 45-49	45-46, 45-49	126.4	-	126.2	26-30, 45-46 42-49
Outage 5: 49-66	-	-	126.4	-	126.2	26-30, 45-46 45-49
Outage 6: 66-67	64-65, 65-66	64-65, 65-66	126.7	64-65	126.2	26-30, 45-46, 45-49, 64-65, 65-66

There could be situations when the first component takes longer time to generate a solution. Under such circumstances, the second component should be used (as discussed in Section 4.4). Table 5.2 presents the application of the second component and compares it with a simple DC-OPF. Note that it is fair to compare the second component with a DC-OPF instead of an AC-OPF because the DC-OPF solves a linearized constrained optimization problem (similar to rCA used in the second component) while the optimization problem solved in AC-OPF is non-linear. Moreover, the focus here is on high-speed, and it is well-known that for a given system, a DC-OPF problem can be solved much faster than an AC-OPF problem.

The first column of Table 5.2 lists the sequence of events. Columns two through five present the results of the second component. Note that only the FT results are shown in this section as the RTCA results are not considered in the second component. Cascading

analysis done after the corrective action indicates that the redispatch obtained from rCA does not contain any cascade triggering contingency for the first five consecutive outages (see fifth column of Table 5.2). However, after the sixth outage, two cascade triggering contingencies manifest before the corrective action is initiated (see last row, third column of Table 5.2), of which, only one is addressed by rCA. That is, the solution obtained using the rCA still contains one cascade triggering contingency (see last row, fifth column of Table 5.2). This happened because the contingency 64-65 triggered cascading failures due to branch overloads, even after the rCA alleviated all post-contingency cut-set saturation.

However, the second component performs significantly better than a DC-OPF (see columns six and seven of Table 5.2). The sixth column presents the DC-OPF redispatch solution, while the seventh column presents the cascading analysis results after the corrective action has been implemented. Since a DC-OPF does not model any security constraints, the number of cascade triggering contingencies in the solution is significantly higher compared to that obtained using rCA (in the second component). *This shows that in situations when the first component takes a long time to generate a solution due to heavy computational burden, the second component can be used to secure the system against post-contingency cut-set saturation, and thereby reduce the risk of cascading failures*.

5.1.2 Mitigation of Saturated Cut-sets Considering Different Asset ratings

The proposed first and the second components are generic enough to initiate corrective actions considering transmission asset ratings determined by different criteria. To demonstrate the application of the first and second components two different scenarios are considered: (a) *Scenario 1*: asset ratings with 95% of the normal value, and (b) *Scenario 2*: asset ratings with 105% of the normal value. Same sequence of outages presented in

previous sub-section are considered here. Tables 5.3 and 5.4 present the performance of the proposed first and second components, respectively. It is observed that the generation costs obtained from either the first or second components in Scenario 1 are greater (or equal) than that in Scenario 2 for every outage (compare the fourth and seventh columns of Table 5.3, and the third and fifth columns of Table 5.4). This is expected because of the more conservative asset ratings of Scenario 1 as compared to Scenario 2.

Table 5.3 Performance of the First Component (FT-RTCA-iCA) Considering Different Asset Ratings During Multiple Outages in the IEEE 118-bus Test System

Event (branch outages)	Scenario 1: Rating: 95%×Normal			Scenario 2: Rating: 105%×Normal		
	RTCA	FT	iCA: Gen. Cost (k$)	RTCA	FT	iCA: Gen. Cost (k$)
Outage 1: 15-33	8-5	-	126.3	-	-	126.2
Outage 2: 19-34	8-5	-	126.3	8-5	-	126.2
Outage 3: 37-38	42-49 64-65 8-5 47-69 26-30 49-69 63-64 63-59	42-49	126.7	42-49 26-30	42-49	126.4
Outage 4: 42-49	45-46 45-49	45-46 45-49	126.8	45-46 45-49	45-46 45-49	126.5
Outage 5: 49-66	8-5	-	126.8			126.5
Outage 6: 66-67	64-65	64-65 65-66	127.4	64-65	65-66	126.9

Table 5.4 Performance of the Second Component (FT-rCA) Considering Different Asset Ratings During Multiple Outages in the IEEE 118-bus Test System

Event (branch outages)	Scenario 1: Rating: 95%×Normal		Scenario 2: Rating: 105%×Normal	
	FT	rCA: Gen. cost (k$)	FT	rCA: Gen cost (k$)
Outage 1: 15-33	-	126.2	-	126.2
Outage 2: 19-34	-	126.2	-	126.2
Outage 3: 37-38	42-49	126.3	42-49	126.2
Outage 4: 42-49	45-46 45-49	126.5	45-46 45-49	126.3
Outage 5: 49-66		126.5		126.3
Outage 6: 66-67	64-65 65-66	126.9	65-66	126.6

5.1.3 Application of the Proposed Methodology to Different Case-studies

To validate the consistency of the first and second components, 40 different case-studies were generated (in addition to the case-study presented in detail in Section 5.1.1). To produce critical scenarios, multiple successive outages were created in different regions of the system. The list of all case-studies is presented in Appendix C of the dissertation. The pseudocodes of the proposed first and second components are present in Appendix D and E of the dissertation, respectively.

The number of successive outages varied between two to six for different case-studies (among the forty-one case-studies, twelve, fifteen, eleven, one, and two case-studies contained 2, 3, 4, 5, and 6 successive outages, respectively). The redispatch solution obtained from the proposed (first and second components) and traditional (RTCA-SCED

and DC-OPF) approaches were evaluated by MATCASC [76] to check if the solution contained cascading contingencies for any of the outages involved in the case-study.

As the computation time of the first component and the traditional RTCA-SCED framework are of similar order (verified experimentally in Section 5.2.2), their performance, denoted by bars with A and B markers, respectively, in Fig. 5.1, were compared first. It is observed from the figure that the redispatch solution from RTCA-SCED contained cascade triggering contingencies for case-studies involved with three (1), four (2), and six (1) outages. However, when the first component was used, none of the case-studies contained any cascade triggering contingencies (bar A is absent in Fig. 5.1).

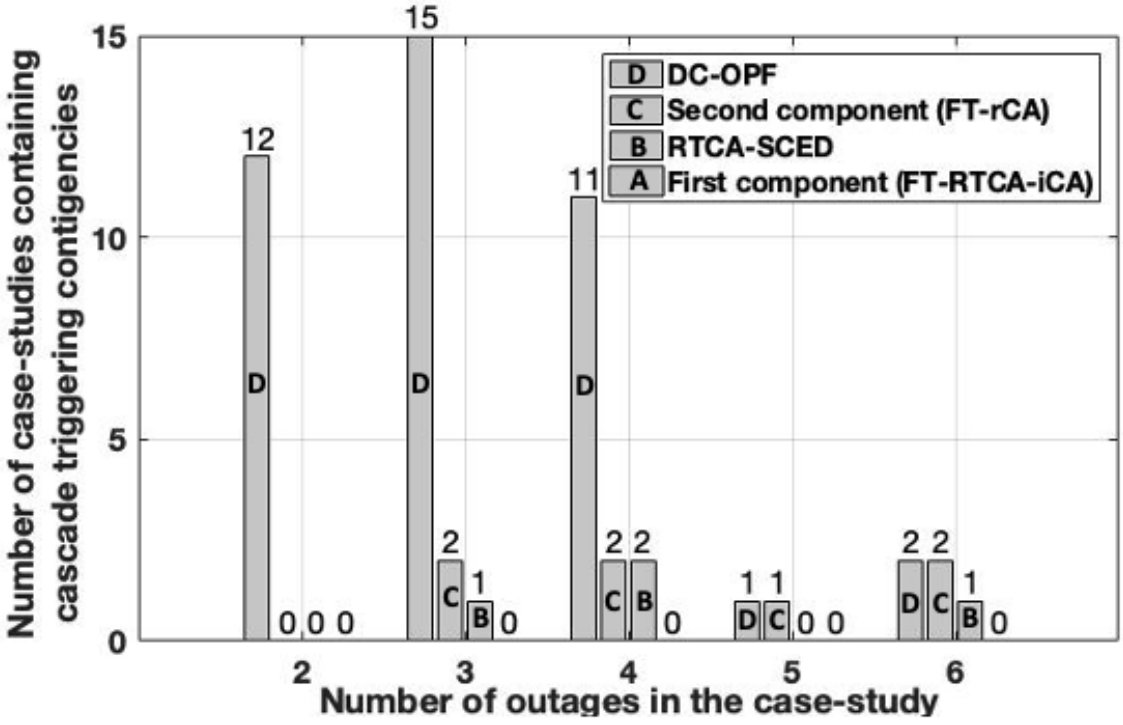

Fig. 5.1 Statistical Summary of Performance of Different Approaches for 41 Case-studies in the IEEE 118-bus Test System

Owing to the similar computation time of the second component and DC-OPF (verified experimentally in Section 5.2.2), their performance, denoted by bars C and D, respectively, in Fig. 5.1, were compared next. It is observed from the figure that the redispatch

solution from DC-OPF contained cascade triggering contingencies for all the case-studies. This is because a DC-OPF does not model any security constraints. However, when the second component was used, the number of case-studies containing cascade triggering contingencies decreased considerably in comparison to the DC-OPF results (compare the heights of bars C and D in Fig. 5.1). *This statistical comparison confirms that during multiple outage scenarios, the proposed two-component methodology can lower, if not eliminate, the risk of cascade triggering contingencies in comparison to traditional approaches*.

5.2 Mitigation of Saturated Cut-sets in the 2000-bus Synthetic Texas System

First, the solution quality of the two-component methodology is compared with traditional approaches such as RTCA-SCED and DC-OPF for a specific case-study of the 2000-bus synthetic Texas system [112]. Finally, based upon the computation time, the real-time applicability of the two-components is explained for the same case-study. The total power demand in the system is 67,109 MW.

5.2.1 A Detailed Case-study of the 2000-bus Synthetic Texas System

In this section, we first explain the performance of the first component (FT-RTCA-iCA) against traditional RTCA-SCED when a sequence of three successive outages manifest on the 2,000-bus synthetic Texas system. The first column of Table 5.5 lists the events which occur successively in the system. The second column shows the number of special assets ($|E_s|$) detected by the FT algorithm. The third column presents the number of critical contingencies ($|E_v|$) detected by RTCA that will create post-contingency branch overloads. To determine the entries of this column, contingency ranking was performed (using PTDFs

and asset ratings [53]), following which top 30% of the contingencies were evaluated by RTCA [27]. The fourth and the seventh column presents the number of critical contingencies detected by MATCASC *before* and *after* the corrective action (denoted as $|E_c|$ and $|E_c'|$) respectively. The fifth column shows the number of critical contingencies detected by RTCA and FT, which are common to the cascade triggering contingencies obtained from MATCASC (denoted as$|E'' \cap E_c|$, where $E'' = E_s \cup E_v$). The sixth column presents the redispatch solution (generation cost, load shed) obtained from the iCA. *Finally, we observe from the last column of Table 5.5 that the solution from iCA (in the first component) does not contain any cascade triggering contingencies for any of the three outages.*

Table 5.5 Performance of the First Component (FT-RTCA-iCA) on the 2000-bus Synthetic Texas System During a Sequence of Outages

Event	FT $\|E_s\|$	RTCA $\|E_v\|$	MATCASC (before correction) $\|E_c\|$	$\|E'' \cap E_c\|$	iCA		MATCASC (after correction) $\|E_c'\|$
					Gen. Cost (k$)	Load-shed (MW)	
Outage 1 (3047-3129)	3	2	3	3	922.4	106	0
Outage 2 (1004-3133)	8	10	6	6	924.7	0	0
Outage 3 (3127-3141)	7	7	12	8	930.8	0	0

Table 5.6 Performance of the RTCA-SCED on the 2,000-bus Synthetic Texas System During a Sequence of Outages

Event	RTCA $\|E_v\|$	MATCASC (before correction) $\|E_c\|$	$\|E_v \cap E_c\|$	SCED		MATCASC (after correction) $\|E_c'\|$
				Gen. Cost (k$)	Load- shed (MW)	
Outage 1 (3047-3129)	2	3	1	922.7	106	0
Outage 2 (1004-3133)	10	6	2	924.6	0	2
Outage 3 (3127-3141)	7	15	2	929.9	0	1

The performance of the traditional RTCA-SCED is presented in Table 5.6. The column headings of Table 5.6 are similar to that of the first component, with the only difference that FT results are not presented because the SCED does not use inputs from the FT. The last column of Table 5.6 shows that the redispatch solution from the SCED contains two and one cascade triggering contingency after the second and third outages, respectively. This becomes clear when the fifth column of Table 5.5 is compared with the fourth column of Table 5.6: *the* $|E_v \cap E_c|$ *of Table 5.6 is significantly less than* $|E'' \cap E_c|$ *of Table 5.5 for every outage*. This shows the RTCA alone is not able to identify a larger proportion of the cascade triggering contingencies, and consequently the SCED is unable to mitigate all cascading failures. On the other hand, *identification of saturated cut-sets by FT, critical branch overloads by RTCA and joint modeling of these constraints within iCA (of the first component) minimizes the risk of cascading failures.*

Now, the performance of the second component (FT-rCA) is evaluated in detail for the same sequence of outages. The column headings of Table 5.7 are similar to that of Table 5.5 with the only difference that the RTCA results are not reported in Table 5.7. This is primarily because the rCA does not utilize inputs from RTCA. However, it is interesting to note that FT alone detects a significant number of the cascade triggering contingencies before the corrective action is implemented (compare the third and fourth columns of Table 5.7). Moreover, it is observed that $|E_s \cap E_c|$ of Table 5.7 is higher compared to $|E_v \cap E_c|$ of Table 5.6. *This indicates that the special assets detected by FT, whose potential outage saturates a cut-set in the network are more likely to trigger cascading failures in the system*. The redispatch solution obtained from the rCA is shown in the fifth column of Table

5.7. Finally, it is observed from the last column that for the listed sequence of outages, that the rCA does not contain any cascade triggering contingency.

The second component performs significantly better than a DC-OPF (see Table 5.8). The second column presents the DC-OPF redispatch solution, while the third column presents the cascading analysis results after the corrective action has been implemented. Comparing the last columns of Table 5.7 and Table 5.8, we observe that the number of cascade triggering contingencies in the DC-OPF solution is significantly higher compared to that obtained using the rCA (in the second component). This is primarily because a DC-OPF does not model any security constraints.

Table 5.7 Performance of the Second Component (FT-rCA) on the 2000-bus Synthetic Texas System During a Sequence of Outages

Event	FT $\lvert E_s \rvert$	MATCASC (before correction) $\lvert E_c \rvert$	$\lvert E_s \cap E_c \rvert$	rCA		MATCASC (after correction) $\lvert E_c' \rvert$
				Gen. Cost (k$)	Load-shed (MW)	
Outage 1 (3047-3129)	3	3	3	917.8	0	0
Outage 2 (1004-3133)	8	6	6	922.5	0	0
Outage 3 (3127-3141)	7	18	7	925.2	0	0

Table 5.8: Performance of DC-OPF on the 2000-bus Synthetic Texas System During a Sequence of Outages

Event	DC-OPF		MATCASC (after correction) $\lvert E_c' \rvert$
	Gen. cost (k$)	Load-shed (MW)	
Outage 1 (3047-3129)	915.9	0	3
Outage 2 (1004-3133)	915.9	0	9
Outage 3 (3127-3141)	917.9	0	21

5.2.2 The Computation Times of Different Approaches

Let us consider the same sequence of three successive outages in this system (as shown in the first column of Table 5.9). The second, third, fourth, and fifth columns present the computation time of traditional RTCA-SCED, first component, DC-OPF, and second component, respectively. It can be observed from the second and third columns that the computation times of RTCA-SCED and the first component are of similar order. This is because the computational speeds of both of these approaches depend heavily on the number of critical contingencies identified by RTCA. This becomes especially clear after the third outage occurs (see last row, second and third columns of Table 5.9). After this (third) outage, a relatively large number of violations were modeled as post-contingency branch overload constraints of SCED and iCA, which consequently increased the computation time of the traditional RTCA-SCED and the first component, respectively. It must also be noted that for this system, the computation time for SCED and iCA were obtained after the PTDFs lower than 0.02 were rounded off to 0. When this rounding was not done, due to the extremely high computational burden of the optimization problem for RTCA-SCED and the first component, the local memory of the solver became insufficient.

Table 5.9 Time Comparisons of Different Approaches During a Sequence of Outages on the 2000-bus Synthetic Texas System

Event	Time*			
	RTCA-SCED	First component (FT-RTCA-iCA)	DC-OPF	Second component (FT-rCA)
Outage 1: 3047-3129	388 sec	421 sec	15 sec	28 sec
Outage 2: 1004-3133	431 sec	487 sec	20 sec	21 sec
Outage 3: 3127-3141	622 sec	720 sec	24 sec	20 sec

*The simulations were performed on a computer with 2.3 GHz Dual-Core Intel Core i5 processor and 8 GB RAM.

On a similar note, the computation times of DC-OPF and the second component are found to be very similar (see fourth and fifth columns of Table 5.9). Both were less than 30 seconds for this system, which is at least an order of magnitude faster than the first component and RTCA-SCED. The high speed is primarily because the DC-OPF and rCA (used in the second component) do not model the computationally intensive post-contingency branch overload constraints. Furthermore, it is important to note that the optimization problems of the rCA and DC-OPF do not require any approximation of the PTDFs. However, the performance of the second component is superior in comparison to a simple DC-OPF because the former incorporates a relaxed criterion of security using the cut-set power transfer constraints (modeled inside rCA). *Thus, the rCA is able to provide security against post-contingency cut-set saturation without significantly increasing the computational burden of the resulting optimization problem.*

5.2.3 Real-time Implementation of the Proposed Methodology

Table 5.10 presents the real-time application of the two-component methodology for the three outages described in the previous sub-section. The first column lists the sequence of events. Let us assume that for this system the redispatch must be implemented every 10 minutes. Keeping this in mind, it can be observed from Table 5.9 that the first component yields a result within 10 minutes for the first two outages, whereas the computation time increases beyond 10 minutes after the third outage. Therefore, the redispatch solution from the first component should be implemented after the first and second outages occur, whereas the results from the second component should be used for redispatch after the third outage (as mentioned in the second column, last row of Table 5.10).

The third column presents the solution (generation cost and load-shed) obtained when one of the two components of the proposed methodology is implemented after every outage to mitigate the identified post-contingency violations. A summary of the observations made from the dispatch solution in Table 5.10 is provided below.

Table 5.10 Real-time Application of the Two-component Methodology During a Sequence of Outages on the 2000-bus Synthetic Texas System

Events	Method	Dispatch Solution		No. of cascade triggering contingencies detected by MATCASC (after correction)
		Gen. cost (k$)	Load-shed (MW)	
Outage 1: 3047-3129	First component: FT-RTCA-iCA	922.4	106	0
Outage 2: 1004-3133	First component: FT-RTCA-iCA	924.7	0	0
Outage 3: 3127-3141	Second component: FT-rCA	923.2	0	0

- *Outage 1:* The generation redispatch (obtained using the first component) alone cannot mitigate the identified post-contingency violations. Therefore, 106 MW of load is shed at this stage. Therefore, the remaining load in the system becomes 67,003 (= 67,109-106) MW. The total generation fleet satisfies the power demand of 67,003 MW at the generation cost of $ 922.4k.
- *Outage 2:* Following the second event, the first component is implemented once more. To mitigate additional post-contingency violations, the generation cost for redispatch increases to $ 924.7k. The redispatch solution involves no additional load-shed, and so the load of 67,003 MW is satisfied by the new generation dispatch.
- *Outage 3:* Following the third event, the second component is implemented. The redispatch solution involves no additional load-shed indicating that the total generation now satisfies the power demand of 67,003 MW at a new generation cost of $ 923.2k. Note

that the slight decrease in the generation cost from $ 924.7k to $ 923.2k is due to the relaxed security constraints of rCA (in the second component) compared to the more conservative security constraints of iCA (in the first component).

Finally, the last column presents the number of cascade-triggering contingencies contained in the solution. It is observed that for the listed sequence of events, the solution obtained from the proposed methodology does not contain any cascade triggering contingencies. *Therefore, this case-study illustrates the real-time implementation of the two components in large power systems during multiple outages.*

CHAPTER 6

CONCLUSION

This Chapter summarizes the research findings and contributions of this dissertation. Further, it introduces different research problems that could be investigated in future works by building upon this research.

6.1 Dissertation Summary

This dissertation first proposes a new graph-theoretic approach for real-time security assessment in large power systems for enhanced situational awareness. The most important research finding is that *a relaxed graph theory-based network analysis tool can efficiently analyze if a contingency will create saturated cut-sets in a meshed power system.* The proposed feasibility test (FT) algorithm utilizes exhaustive graph traversal using the breadth first search (BFS) technique to determine post-contingency cut-set saturation. Identification of saturated cut-sets is important, because they are the "vulnerable bottlenecks in power grids and represent seams or fault lines across which islanding seems likely" [92]. However, any large power networks can be associated with countless number of cut-sets. In this context, the unique contribution of the FT algorithm can be stated as follows: *the FT can quickly analyze the power transfer capability across all cut-sets (without the cut-sets being pre-defined), and uniquely detect saturated cut-sets due to a potential contingency in the system*.

Computation speed is an important criterion for real-time power system operations. To enhance the computational efficiency of the FT algorithm after a branch outage or generation redispatch in the power system additional graph-theoretic algorithms were

developed. The update scheme (UPS) and shortlisting assets (SA) algorithm increases the computation speed of FT following a branch outage in the system, whereas the modified update scheme (M-UPS) and modified shortlisting assets (M-SA) algorithm provides the necessary computational boost following a generation redispatch.

Finally, going beyond the detection of saturated cut-sets in power systems, this dissertation demonstrates how the power system can be made secure against post-contingency cut-set saturation using a combination of network science and constrained optimization. A two-component methodology is developed to enhance the *N*-1 security during successive outages in power systems. The first component of the proposed methodology combines the results from the FT algorithm and traditional RTCA to create an integrated corrective action (iCA). The iCA initiates a comprehensive response to the violations detected by FT and RTCA to protect the system against saturated cut-sets as well as critical branch overloads. The second component of the proposed methodology presents an alternative method that complements real-time power system operations during extreme event scenarios, when detailed network analysis tools such as the first component or traditional RTCA-SCED take longer time to generate a solution. Under such circumstances, by only employing the FT algorithm, a relaxed corrective action (rCA) is implemented that quickly mitigates saturated cut-sets in power systems.

6.2 Future Work

Any research work paves the path for more studies that can be done and many research questions that can be explored along similar lines. The findings of this dissertation have also led to exciting research questions that could be investigated in the future. A

summary of the different avenues that could be explored building upon this research are summarized below.

Visualization of electric power transmission systems is important for supporting the study, analysis, and presentation of power system data [113]-[115]. An immediate application of the proposed algorithms could be the development of a robust visualization software to automatically display saturated cut-sets (detected by the FT) in the context of the local geography when outages manifest successively in a region. Automated visualization of the information captured by the FT would facilitate quick and easy situational awareness of the detected violations in a power systems control room. Recently, in [113], the authors proposed a new algorithm for efficient automatic visualization of large power systems, that merges geographical context with logical clarity. These techniques can be explored for the development of high-end visualization platforms for the research done in this dissertation.

In power systems there may exist some practical constraints with regards to the total power transfer capacity for a set of transmission lines based upon a contractual agreement (or rules) between different utilities [116]-[117]. Detection of post-contingency cut-set saturation with the consideration of these additional constraints can be explored in the future.

Quick detection and mitigation of saturated cut-sets due to *transmission* contingencies has been the focus of this dissertation. The impact of generator contingencies on different cut-sets is an interesting research problem as well. Following a generator contingency, the deficient power is picked up by other generators in the system based upon the generator shift factors [118]-[121]. The redistribution of power flows following a generator

contingency can also create saturated cut-sets (or bottlenecks) in power networks. Building on the proposed FT algorithm, intelligent graph traversal techniques can be developed to detect saturated cut-sets due to a potential generator contingency. At the same time, all generators are not likely to respond to generator contingencies; for example, nuclear power plants are always operated on their baseload and do not respond to frequency events [122]. Such realistic assumptions should be explored for faster detection of saturated cut-sets due to generator contingencies.

Generation redispatch and controlled load shedding has been used as corrective measures to alleviate post-contingency cut-set saturation in this dissertation. Research on transmission switching [27]-[28] and topology re-configuration [123]-[126] for efficient congestion management has also been a promising field of recent research, especially for weakly meshed distribution networks. The prospect of using topology re-configuration (as a corrective action) to mitigate post-contingency cut-set saturation, can be investigated as an extension of this dissertation.

The corrective actions developed in this dissertation is based upon a simple DC power flow model, which quickly mitigates saturated cut-sets based on active power transfer. As such, the violations associated with the voltage magnitude and reactive power flows are not identified. The possibility of developing more detailed corrective actions with the consideration of voltage magnitude (and reactive power flows), while simultaneously mitigating post-contingency cut-set saturation could be explored in the future. However, the challenge here is the following: *a detailed AC OPF formulation with the consideration of non-linear power flow equations is likely to compromise the computational efficiency of the proposed graph-theoretic algorithms.* Therefore, future research needs to explore the

recent advancements in linearized OPF techniques which involve intelligent approximations in the OPF model but generate results close to the optimum of AC OPF [127]-[128].

The detection and mitigation of saturated cut-sets in power systems has been treated as a steady-state network analysis problem in this dissertation. The redispatch solution (obtained from the proposed corrective actions) alleviates potential violations with regards to the steady-state security of power systems. However, when the generator set-points are changed (corresponding to the new generation dispatch) there is possibility of stability related violations manifesting during power system transients. To ensure that stability related violations do not arise during the transient period of system redispatch, transient stability assessment [129]-[132] and transient stability constrained optimal power flow techniques [133]-[137] could be explored in future work.

Finally, consideration of high renewable energy penetration in power systems will add another level of complexity to the detection and mitigation schemes of post-contingency cut-set saturation. This is because high renewable penetration implies that during generation redispatch we have limited controllability to maneuver generation resources to mitigate saturated cut-sets. Further, if corrective actions are to be initiated with the consideration of the variable renewable generation (in the next dispatch cycle), stochastic optimal power flow techniques [138]-[141] can be explored in the context of the problem being solved in this dissertation.

REFERENCES

[1] A. Abur, and A. G. Exposito, *Power system state estimation: theory and implementation*. Boca Raton, FL, USA: CRC, 2004.

[2] F. C. Schweppe, "Power system static-state estimation, part III: implementation," *IEEE Trans. Power App. Syst.*, vol. PAS-89, no. 1, pp. 130-135, Jan. 1970.

[3] A. G. Phadke, and J. S. Thorp, *Synchronized phasor measurements and their applications*. New York: Springer, 2008.

[4] A. G. Phadke, J. S. Thorp, and M. G. Adamiak, "A new measurement technique for tracking voltage phasors, local system frequency, and rate of change of frequency," *IEEE Trans. Power App. Syst.*, vol. PAS-102, no. 5, pp. 1025-1038, May 1983.

[5] A. G. Phadke, J. S. Thorp, R. F. Nuqui, and M. Zhou, "Recent developments in state estimation with phasor measurements," in *Proc. IEEE/PES Power Syst. Conf. and Expos.*, pp. 1–7, Mar. 2009.

[6] Z. Chu, A. Pinceti, R. Sen Biswas, O. Kosut, A. Pal, and L. Sankar, "Can predictive filters detect gradually ramping false data injection attacks against PMUs?," in *Proc. IEEE Int. Conf. Commun. Control Comput. Technol. Smart Grids (SmartGridComm)*, Beijing, China, pp. 1-6, Oct. 2019.

[7] A. Rouhani, and A. Abur, "Linear phasor estimator assisted dynamic state estimation," *IEEE Trans. Smart Grid*, vol. 9, no. 1, pp. 211-219, Jan. 2018.

[8] T. Yang, H. Sun, and A. Bose, "Transition to a two-level linear state estimator—part I: architecture," *IEEE Trans. Power Syst.*, vol. 26, no. 1, pp. 46-53, Feb. 2011.

[9] T. Yang, H. Sun, and A. Bose, "Transition to a two-level linear state estimator—part II: algorithm," *IEEE Trans. Power Syst.*, vol. 26, no. 1, pp. 54-62, Feb. 2011.

[10] K. D. Jones, J. S. Thorp, and R. M. Gardner, "Three-phase linear state estimation using phasor measurements," in *Proc. IEEE Power Eng. Soc. Gen. Meeting*, Vancouver, BC, Canada, pp. 1-5, Jul. 2013.

[11] P. Chatterjee, A. Pal, J. S. Thorp, and J. De La Ree, "Partitioned linear state estimation," in *Proc. IEEE Power Eng. Soc. Innovative Smart Grid Technol. Conf. (ISGT)*, Washington, DC, USA, pp. 1-5, Feb. 2015.

[12] V. Chakati, M. Pore, A. Banerjee, A. Pal, and S. K. S. Gupta, "Impact of false data detection on cloud hosted linear state estimator performance," in *Proc. IEEE Power Eng. Soc. Gen. Meeting (PESGM)*, Portland, OR, USA, pp. 1-5, Aug. 2018.

[13] B. Azimian, R. Sen Biswas, A. Pal, and L. Tong, "Time synchronized state estimation for incompletely observed distribution systems using deep learning considering realistic

measurement noise," in *Proc. IEEE Power Eng. Soc. Gen. Meeting*, Washington D.C., USA, pp. 1-5, Jul. 2021.

[14] D. A. Haughton, and G. T. Heydt, "A linear state estimation formulation for smart distribution systems," *IEEE Trans. Power Syst.*, vol. 28, no. 2, pp. 1187-1195, May 2013.

[15] K. D. Jones, A. Pal, and J. S. Thorp, "Methodology for performing synchrophasor data conditioning and validation," *IEEE Trans. Power Syst.*, vol. 30, no. 3, pp. 1121-1130, May 2015.

[16] M. Padhee, R. Sen Biswas, A. Pal, K. Basu, and A. Sen, "Identifying unique power system signatures for determining vulnerability of critical power system assets," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 47, no. 4, pp. 8-11, Apr. 2020.

[17] A. Pal, A. K. S. Vullikanti and S. S. Ravi, "A PMU placement scheme considering realistic costs and modern trends in relaying," *IEEE Trans. Power Syst.*, vol. 32, no. 1, pp. 552-561, Jan. 2017.

[18] A. Pal, G. A. Sanchez-Ayala, V. A. Centeno, and J. S. Thorp, "A PMU placement scheme ensuring real-time monitoring of critical buses of the network," *IEEE Trans. Power Del.*, vol. 29, no. 2, pp. 510-517, Apr. 2014.

[19] A. Pal, C. Mishra, A. K. S. Vullikanti, and S. S. Ravi, "General optimal substation coverage algorithm for phasor measurement unit placement in practical systems," *IET Gener., Transm. Distrib*., vol. 11, no. 2, pp. 347-353, Jan. 2017.

[20] A. Pal, G. A. Sanchez-Ayala, J. S. Thorp, and V. A. Centeno, "A community-based partitioning approach for phasor measurement unit placement in large systems," *Elect. Power Compon. Syst*., vol. 44, no. 12, pp. 1317-1329, Jun. 2016.

[21] C. Mishra, K. D. Jones, A. Pal, and V. A. Centeno, "Binary particle swarm optimisation-based optimal substation coverage algorithm for phasor measurement unit installations in practical systems," *IET Gener. Transm. Distrib.*, vol. 10, no. 2, pp. 555-562, Feb. 2016.

[22] M. Ghamsari-Yazdel, M. Esmaili, F. Aminifar, P. Gupta, A. Pal, and H. A. Shayanfar, "Incorporation of controlled islanding scenarios and complex substations in optimal WAMS design," *IEEE Trans. Power Syst*., vol. 34, no. 5, pp. 3408-3416, Sep. 2019.

[23] R. Sen Biswas, B. Azimian, and A. Pal, "A micro-PMU placement scheme for distribution systems considering practical constraints," in *Proc. IEEE Power Eng. Soc. Gen. Meeting,* Montreal, Canada, pp. 1-5, Aug. 2020.

[24] G. A. Sanchez, A. Pal, V. A. Centeno, and W. C. Flores, "PMU placement for the central American power network and its possible impacts," in *Proc. IEEE Innovative Smart Grid Technol. Latin America (ISGT LA),* Medellin, Colombia, pp. 1-7, Oct. 2011.

[25] NERC, "Standard TPL-002-0a – system performance following loss of a single bulk electric system element," 2010. [Online]. Available: https://www.nerc.com/files/tpl-002-0a.pdf

[26] New York Independent System Operator, "RTC-RTD convergence study," Dec. 2017.

[27] X. Li, P. Balasubramanian, M. Sahraei-Ardakani, M. Abdi-Khorsand, K. W. Hedman, and R. Podmore, "Real-time contingency analysis with corrective transmission switching," *IEEE Trans. Power Syst.*, vol. 32, no. 4, pp. 2604-2617, Jul. 2017.

[28] X. Li, and K. W. Hedman, "Enhanced energy management system with corrective transmission switching strategy—part I: methodology," *IEEE Trans. Power Syst.*, vol. 34, no. 6, pp. 4490-4502, Nov. 2019.

[29] M. Barkakati, R. Sen Biswas, and A. Pal, "A PMU based islanding detection scheme immune to additive instrumentation channel errors," in *Proc. IEEE North American Power Symp. (NAPS)*, Wichita, KS, Oct. 2019, pp. 1-6.

[30] R. Sen Biswas, and A. Pal, "A robust techno-economic analysis of PMU-based islanding detection schemes," in *Proc. IEEE Texas Power Energy Conf. (TPEC)*, College Station, TX, pp. 1-6, Feb. 2017.

[31] G. S. Vassell, "Northeast Blackout of 1965," *IEEE Power Eng. Review*, vol. 11, no. 1, pp. 4-8, Jan. 1991.

[32] R. Sugarman, "Power/energy: New York City's blackout: A $350 million drain: Ripple effects off the July 13, 1977, lightning stroke cost the public dearly in lost property, services, end income," *IEEE Spectrum*, vol. 15, no. 11, pp. 44-46, Nov. 1978.

[33] D. N. Kosterev, C. W. Taylor, and W. A. Mittelstadt, "Model validation for the August 10, 1996 WSCC system outage," *IEEE Trans. Power Syst.*, vol. 14, no. 3, pp. 967-979, Aug. 1999.

[34] J. F. Hauer, N. B. Bhatt, K. Shah, and S. Kolluri, "Performance of "WAMS East" in providing dynamic information for the North East blackout of August 14, 2003," in *Proc. IEEE Power Eng. Soc. Gen. Meeting,* Denver, CO, vol. 2, pp. 1685-1690, Jun. 2004.

[35] F. Galvan, S. Mandal, and M. Thomas, "Phasor Measurement Units (PMU) instrumental in detecting and managing the electrical island created in the aftermath of hurricane Gustav," in *Proc. IEEE Power Syst. Conf. Expos.* Seattle, WA, pp. 1-4, Mar. 2009.

[36] FERC and NERC staff "Arizona-Southern California outages on September 8, 2011, causes and recommendations," *Federal Energy Regulatory Commission* and *North American Electric Reliability Corporation*, Apr. 2012.

[37] P. M. Chakalian, L. C. Kurtz, and D. M. Hondula, "After the lights go out: household resilience to electrical grid failure following hurricane Irma," *Nat. Hazards Rev.*, vol. 20, no. 4, pp. 1-14, Nov. 2019.

[38] B. A. Carreras, D. E. Newman, and I. Dobson, "North American blackout time series statistics and implications for blackout risk," *IEEE Trans. Power Syst.*, vol. 31, no. 6, pp. 4406-4414, Nov. 2016.

[39] FERC Staff, "The Con Edison power failure of July 13 and 14, 1977," *U.S. Department of Energy Federal Energy Regulatory Commission*, pp. 2-3, 19-20 Jun. 1978.

[40] T. Werho, V. Vittal, S. Kolluri, and S. M. Wong, "Power system connectivity monitoring using a graph theory network flow algorithm," *IEEE Trans. Power Syst.*, vol. 31, no. 6, pp. 4945-4952, Nov. 2016.

[41] M. Panteli, and D. S. Kirschen, "Situational awareness in power systems: Theory, challenges, applications," *Electric Power Syst. Research*, vol. 122, pp. 140-151, 2015.

[42] A. Abedi, L. Gaudard, and F. Romerio, "Review of major approaches to analyze vulnerability in power systems," *Reliability Eng. & System Safety*, vol. 183, pp. 153-172, Nov. 2018.

[43] Z. Li, J. Wang, H. Sun, and Q. Guo, "Transmission contingency analysis based on integrated transmission and distribution power flow in smart grid," *IEEE Trans. Power Syst.*, vol. 30, no. 6, pp. 3356-3367, Nov. 2015.

[44] J. Baranowski, and D. J. French, "Operational use of contingency analysis at PJM," in *Proc. IEEE Power Eng. Soc. Gen. Meeting*, San Diego, CA, USA, pp. 1-4, Jul. 2012.

[45] F. Garcia, N. D. R. Sarma, V. Kanduri, and G. Nissankala, "ERCOT control center experience in using real-time contingency analysis in the new nodal market," in *Proc. IEEE Power Eng. Soc. Gen. Meeting*, San Diego, CA, USA, pp. 1-8, Jul. 2012.

[46] S. Huang, and V. Dinavahi, "Real-time contingency analysis on massively parallel architectures with compensation method," *IEEE Access*, vol. 6, pp. 44519-44530, 2018.

[47] A. J. Wood, B. F. Wollenberg, and G. B. Sheble. *Power generation, operation, and control*. New York: Wiley-Interscience, 2013.

[48] G. Anderson. *Modelling and Analysis of Electric Power Systems*: *Power Flow Analysis, Fault Analysis, Power Systems Dynamics and Stability*. Lecture 227-0526-00, ETH Zurich. [Online]. Available: https://web.archive.org/web/20170215042633/http://www.eeh.ee.ethz.ch/uploads/tx_ethstudies/modelling_hs08_script_02.pdf

[49] B. Stott, and O. Alsac, "Fast Decoupled Load Flow," *IEEE Trans. Power App. Syst.*, vol. PAS-93, no. 3, pp. 859-869, May 1974.

[50] Contingency Analysis-Baseline. [Online]. Available: https://smartgrid.epri.com/Use-Cases/ContingencyAnalysis-Baseline.pdf.

[51] X. Li, "Reliability enhancements for real-time operations of electric power systems," Ph.D. dissertation, Arizona State University, Dec. 2017.

[52] P. Mitra, V. Vittal, B. Keel and J. Mistry, "A systematic approach to N -1-1 analysis for power system security assessment," *IEEE Power Energy Technol. Syst. J.*, vol. 3, no. 2, pp. 71-80, Jun. 2016.

[53] E. Bompard, E. Pons, and D. Wu, "Extended topological metrics for the analysis of power grid vulnerability," *IEEE Syst. J.*, vol. 6, no. 3, pp. 481-487, Sep. 2012.

[54] E. Bompard, R. Napoli, and F. Xue, "Extended topological approach for the assessment of structural vulnerability in transmission networks," *IET Gener. Transm. Distrib.*, vol. 4, no. 6, pp. 716-724, Jun. 2010.

[55] A. K. Srivastava, T. A. Ernster, R. Liu, and V. G. Krishnan, "Graph-theoretic algorithms for cyber-physical vulnerability analysis of power grid with incomplete information," *J. Modern Power Syst. Clean Energy*, vol. 6, no. 5, pp. 887-899, Sep. 2018.

[56] C. M. Davis, and T. J. Overbye, "Multiple element contingency screening," *IEEE Trans. Power Syst.*, vol. 26, no. 3, pp. 1294-1301, Aug. 2011.

[57] T. Guler, G. Gross, and M. Liu, "Generalized line outage distribution factors," *IEEE Trans. Power Syst.*, vol. 22, no. 2, pp. 879-881, May 2007.

[58] T. Guler, and G. Gross, "Detection of island formation and identification of casual factors under multiple line outages," *IEEE Trans. Power Syst.*, vol. 22, no. 2, pp. 505-513, May 2007.

[59] H. Ronellenfitsch, M. Timme, and D. Witthaut, "A dual method for computing power transfer distribution factors," *IEEE Trans. Power Syst.*, vol. 32, no. 2, pp. 1007-1015, Mar. 2017.

[60] P. Kaplunovich, and K. Turitsyn, "Fast and reliable screening of N-2 contingencies," *IEEE Trans. Power Syst.*, vol. 31, no. 6, pp. 4243-4252, Nov. 2016.

[61] T. Werho, V. Vittal, S. Kolluri, and S. M. Wong, "A Potential Island Formation Identification Scheme Supported by PMU Measurements," *IEEE Trans. Power Syst.*, vol. 31, no. 1, pp. 423-431, Jan. 2016.

[62] K. E. Van Horn, A. D. Dominiguez-Garcia, and P. W. Sauer, "Measurement-based real-time security constrained economic dispatch," *IEEE Trans. Power Syst.*, vol. 31, no. 5, pp. 3548-3560, Sep. 2016.

[63] ISO New England Inc. System Planning, "Transmission planning technical guide." [Online]. Available: https://www.iso-ne.com/static-assets/documents/2019/10/transmission_plannings_techincal_guide_rev5.pdf

[64] A. N. Madavan, S. Bose, Y. Guo, and L. Tong, "Risk-sensitive security-constrained economic dispatch via critical region exploration," in *Proc. IEEE Power Energy Soc. Gen. Meeting (PESGM)*, Atlanta, GA, USA, pp. 1-5, Aug. 2019.

[65] H. Guo, C. Zheng, H. H.-C. Iu, and T. Fernando, "A critical review of cascading failure analysis and modeling of power system," *Renew. Sustain. Energy Rev.*, vol. 80, pp. 9-22, Apr. 2017.

[66] R. Baldick, *et al*., "Initial review of methods for cascading failure analysis in electric power transmission systems IEEE PES CAMS task force on understanding, prediction, mitigation and restoration of cascading failures," in *Proc. IEEE Power Eng. Soc. Gen. Meeting - Conversion and Delivery of Electrical Energy in the 21st Century,* Pittsburgh, PA, pp. 1-8, Jul. 2008.

[67] D. S. Kirschen, D. Jayaweera, D. P. Nedic, and R. N. Allan, "A probabilistic indicator of system stress," *IEEE Trans. Power Syst.*, vol. 19, no. 3, pp. 1650-1657, Aug. 2004.

[68] R. Hardiman, M. Kumbale, and Y. Makarov, "An advanced tool for analyzing multiple cascading failures," in *Proc. Int. Conf. Prob. Methods Applied Power Syst.*, Ames, IA, pp. 629-634, Sep. 2004.

[69] B. A. Carreras, V. E. Lynch, I. Dobson, and D. E. Newman, "Complex dynamics of blackouts in power transmission systems," *Chaos*, vol. 14, no. 3, pp. 643-652, Sep. 2004.

[70] J. Chen, J. S. Thorp, and I. Dobson, "Cascading dynamics and mitigation assessment in power system disturbances via a hidden failure model," *Int. J. Electrical Power Energy Syst.*, vol. 27, no. 4, 2005.

[71] I. Dobson, B. A. Carreras, D. E. Newman, and J. M. Reynolds-Barredo, "Obtaining statistics of cascading line outages spreading in an electric transmission network from standard utility data," *IEEE Trans. Power Syst.*, vol. 31, no. 6, pp. 4831-4841, Nov. 2016.

[72] I. Dobson, "Estimating the propagation and extent of cascading line outages from utility data with a branching process," *IEEE Trans. Power Syst.*, vol. 27, no. 4, pp. 2146-2155, Nov. 2012.

[73] P. Rezaei, P. D. H. Hines, and M. J. Eppstein, "Estimating cascading failure risk with random chemistry," *IEEE Trans. Power Syst.*, vol. 30, no. 5, pp. 2726-2735, Sep. 2015.

[74] M. Rahnamay-Naeini, Z. Wang, N. Ghani, A. Mammoli, and M. M. Hayat, "Stochastic analysis of cascading-failure dynamics in power grids," *IEEE Trans. Power Syst.*, vol. 29, no. 4, pp. 1767-1779, Jul. 2014.

[75] P. D. H. Hines, I. Dobson, and P. Rezaei, "Cascading power outages propagate locally in an influence graph that is not the actual grid topology," *IEEE Trans. Power Syst.*, vol. 32, no. 2, pp. 958-967, Mar. 2017.

[76] Y. Koç, T. Verma, N. A. M. Araujo, and M. Warnier, "MATCASC: A tool to analyse cascading line outages in power grids," in *Proc. IEEE Intl. Workshop Intelligent Energy Syst. (IWIES),* Vienna, Austria, pp. 143-148, Nov. 2013.

[77] T. Ishizaki, A. Chakrabortty, and J. Imura, "Graph-theoretic analysis of power systems," in *Proc. IEEE,* vol. 106, no. 5, pp. 931-952, May 2018.

[78] R. Albert, I. Albert, and G. L. Nakarado, "Structural vulnerability of the North American power grid," *Phys. Rev. E*, vol. 69, pp. 1–10, Feb. 2004.

[79] H. Bai, and S. Miao, "Hybrid flow betweenness approach for identification of vulnerable line in power system," *IET Gener. Transm. Distrib.*, vol. 9, no. 12, pp. 1324-1331, Jan. 2015.

[80] K. Wang, et. al. "An electrical betweenness approach for vulnerability assessment of power grids considering the capacity of generators and loads," *Phys. A: Statist. Mech. Appl.,* vol. 390, no. 23/24, pp. 4692-4701, Nov. 2011.

[81] S. Arianos, E. Bompard, A. Carbone, and F. Xue, "Power grid vulnerability: A complex network approach," *Chaos,* vol. 19, no. 01199, 2009.

[82] P. Crucitti, V. Latora, and M. Marchiori, "Locating critical lines in high-voltage electrical power grids," *Fluctuation Noise Lett.*, vol. 5, no. 2, 2005.

[83] V. Rosato, S. Bologna, and F. Tiriticco, "Topological properties of high-voltage electrical transmission networks," *Electric Power Systems Research*, vol. 77, iss. 2, pp. 99-105, Feb. 2007.

[84] P. Crucitti, V. Latora, and M. Marchiori, "Topological analysis of the Italian electric power grid," *Physica A*, vol. 338, pp. 92-97, 2004.

[85] Y. Koc, M. Warnier, R. E. Kooij, and B. M. T. Frances, "Structural vulnerability assessment of electric power grids," *Electr. Power Syst. Res.*, vol. 81, pp. 1334-1340, 2011.

[86] P. Chopade, and M. Bikdesh, "New centrality measures for assessing smart grid vulnerabilities and predicting brownouts and blackouts," *Int. J. Critical Infrastructure Protection*, vol. 12, pp. 29-45, 2016.

[87] J. Fang, C. Su, Z. Chen, H. Sun, and P. Lund, "Power system structural vulnerability assessment based on an improved maximum flow approach," *IEEE Trans. Smart Grid*, vol. 9, no. 2, pp. 777-785, Mar. 2018.

[88] G. J. Correa-Henao, and J. M. Yusta-Loyo, "Representation of electric power systems by complex networks with applications to risk vulnerability assessment," *DYNA*, vol. 82, no. 192, pp. 68-77, Aug. 2015.

[89] J. Beyza, E. Garcia-Paricio, and J. M. Yusta, "Applying complex network theory to the vulnerability assessment of interdependent energy infrastructures," *Energies*, vol. 12, no. 3, Jan. 2019.

[90] Y. Zhu, J. Yan, Y. Sun, and H. He, "Revealing cascading failure vulnerability in power grids using risk-graph," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 12, pp. 3274–3284, Dec. 2014.

[91] J. Beyza, J. Yusta M, G. Correa J, and H. Ruiz F, "Vulnerability assessment of a large electrical grid by new graph theory approach," *IEEE Latin America Trans.*, vol. 16, no. 2, pp. 527-535, Feb. 2018.

[92] A. Beiranvand, and P. Cuffe, "A topological sorting approach to identify coherent cut-sets within power grids," *IEEE Trans. Power Syst.*, vol. 35, no. 1, pp. 721-730, Jan. 2020.

[93] R. Sen Biswas, A. Pal, T. Werho, and V. Vittal, "A graph theoretic approach to power system vulnerability identification," *IEEE Trans. Power Syst*., vol. 36, no. 2, pp. 923-935, Mar. 2021.

[94] R. Sen Biswas, A. Pal, T. Werho, and V. Vittal, "Fast identification of saturated cut-sets using graph search techniques," in *Proc. IEEE Power Eng. Soc. General Meeting*, Montreal, Canada, pp. 1-5, Aug. 2020.

[95] D. Angel, "A breadth first search approach for minimum vertex cover of grid graphs," in *Proc. IEEE 9th Int. Conf. on Intelligent Syst. and Control (ISCO)*, Coimbatore, pp. 1-4, Jan. 2015.

[96] J. Allen, and F. T. Leighton, "Depth first search and dynamic programming algorithms for efficient CMOS cell generation," in *Advanced Research in VLSI: in Proc. of the Fifth MIT Conf.*, MITP, 1988.

[97] A. Elmasry, and A. Shokry, "A new algorithm for the shortest path problem", *Networks*, vol. 74, pp. 16-39, Dec. 2018.

[98] N. A. Ojekudo, and N. P. Akpan, "An application of Dijkstra algorithm to shortest route problem", *IOSR J. Mathematics*, vol. 13, no. 3, Jun. 2017.

[99] F. Ahmed, F. Anzum, M. N. Islam, W. Mohammad Abdullah, S. A. Ahsan, and M. Rana, "A new algorithm to compute single source shortest path in a real edge weighted graph to optimize time complexity," in *Proc. IEEE/ACIS 17th Int. Conf. Computer and Information Science (ICIS)*, Singapore, pp. 185-191, Jun. 2018.

[100] M. Barbehenn, "A note on the complexity of Dijkstra's algorithm for graphs with weighted vertices," *IEEE Trans. Computers*, vol. 47, no. 2, pp. 263-, Feb. 1998.

[101] L. Luo, M. Wong, and W. Hwu, "An effective GPU implementation of breadth-first search," in *Proc. Design Automation Conf.*, Anaheim, CA, pp. 52-55, Jun. 2010.

[102] R. D. Zimmerman, "MATPOWER 4.0b4 user's manual," [Online]. Available: https://matpower.org/.

[103] D. A. Douglass *et al*., "A review of dynamic thermal line rating methods with forecasting," *IEEE Trans. Power Del.*, vol. 34, no. 6, pp. 2100-2109, Dec. 2019.

[104] IEEE PES Overhead Lines Subcommittee WG 15.11, "Real-time overhead transmission line monitoring for dynamic line rating," *IEEE Trans. Power Del.*, vol. 31, no. 3, pp. 921-929, Jun. 2016.

[105] CIGRE WG B2.36, *Guide for Application of Direct Real-Time Monitoring Systems*. Paris: CIGRE, Technical Brochure 498, Jun. 2012.

[106] "Methods for real-time thermal monitoring of conductor temperature" Electra N° 197, Aug. 2001.

[107] California ISO Technical Bulletin, "Comparison of lossy versus lossless shift factors in the ISO market optimizations", Jun. 2009.

[108] X. Xu, H. Zhang, C. Li, Y. Liu, W. Li, and V. Terzija, "Optimization of the event-driven emergency load-shedding considering transient security and stability constraints," *IEEE Trans. Power Syst.*, vol. 32, no. 4, pp. 2581-2592, Jul. 2017.

[109] N. G. Singhal, N. Li, and K. W. Hedman, "A reserve response set model for systems with stochastic resources," *IEEE Trans. Power Syst.*, vol. 33, no. 4, pp. 4038-4049, Jul. 2018.

[110] PJM, "Security Constrained Economic Dispatch System (SCED)", *Interconnection Training Program*, Module LS 8, Winter 2011. [Online] Available: https://pjm.com/~/media/training/nerc-certifications/ls8-SCED.ashx.

[111] X. Ma, Y. Chen, and J. Wan, "Midwest ISO co-optimization based real-time dispatch and pricing of energy and ancillary services," in *Proc. IEEE Power Eng. Soc. Gen. Meeting*, Calgary, AB, pp. 1-6, Jul. 2009.

[112] A. B. Birchfield, T. Xu, K. M. Gegner, K. S. Shetye, and T. J. Overbye, "Grid structural characteristics as validation criteria for synthetic networks," *IEEE Trans. Power Syst.*, vol. 32, no. 4, pp. 3258-3265, Jul. 2017.

[113] A. B. Birchfield, and T. J. Overbye, "Techniques for Drawing Geographic One-Line Diagrams: Substation Spacing and Line Routing," *IEEE Trans. Power Syst.*, vol. 33, no. 6, pp. 7269-7276, Nov. 2018.

[114] H. Chen, B. Bhargava, F. Habibi-Ashrafi, J. S. Park, and J. Castaneda, "Integration of RTDS with EPG synchrophasor applications for visualization and analysis of simulation scenarios at Southern California Edison," in *Proc. 2012 North American Power Symp. (NAPS)*, Champaign, IL, USA, pp. 1-5, Sep. 2012.

[115] A. Pal, I. Singh, and B. Bhargava, "Stress assessment in power systems and its visualization using synchrophasor based metrics," in *Proc. IEEE 2014 North American Power Symp. (NAPS)*, Pullman, WA, pp. 1-6, Sep. 2014.

[116] W. Hogan, "Contract networks for electric power transmission," *J. Regulatory Econ.*, pp. 211-242, Sep. 1992.

[117] R. D. Christie, B. F. Wollenberg, and I. Wangensteen, "Transmission management in the deregulated environment," in *Proc. IEEE Special Issue The Technol. Power Syst. Competition*, vol. 88, no. 2, pp. 170–195, Feb. 2000.

[118] PJM, "Generator contingency analysis," 2016. [Online] Available: https://www.pjm.com/-/media/training/nerc-certifications/markets-exam-materials/mkt-optimization-wkshp/generator-contingency-analysis.ashx

[119] CAISO, "Draft final proposal: generator contingency and remedial action scheme modeling," Jul. 2017. [Online] Available: https://www.caiso.com/Documents/DraftFinalProposal-GeneratorContingencyandRemedialActionSchemeModeling_updatedjul252017.pdf

[120] N. G. Singhal, and K. W. Hedman, "Generator contingency modeling in electric energy markets." EPRI, 2018. [Online] Available: https://www.ferc.gov/sites/default/files/2020-08/W3B-2_Singhal.pdf

[121] N. G. Singhal, J. Kwon, and K. W. Hedman, "Generator contingency modeling in electric energy markets: derivation of prices via duality theory", 2019, arXiv:1910.02323.

[122] H. Yuan, R. Sen Biswas, J. Tan, and Y. Zhang, "Developing a reduced 240-bus WECC dynamic model for frequency response study of high renewable integration," in *Proc. IEEE/PES Trans. Distrib. Conf. Expos. (T&D)*, Chicago, IL, USA, pp. 1-5, Oct. 2020.

[123] E. A. Goldis, P. A. Ruiz, M. C. Caramanis, X. Li, C. R. Philbrick, and A. M. Rudkevich, "Shift factor-based SCOPF topology control MIP formulations with substation configurations," *IEEE Trans. Power Syst.*, vol. 32, no. 2, pp. 1179-1190, Mar. 2017.

[124] M. Heidarifar, and H. Ghasemi, "A network topology optimization model based on substation and node-breaker modeling," *IEEE Trans. Power Syst.*, vol. 31, no. 1, pp. 247-255, Jan. 2016

[125] S. Huang, Q. Wu, L. Cheng, and Z. Liu, "Optimal reconfiguration-based dynamic tariff for congestion management and line loss reduction in distribution networks," *IEEE Trans. Smart Grid*, vol. 7, no. 3, pp. 1295-1303, May 2016.

[126] S. Pandey, S. Chanda, A. K. Srivastava, and R. O. Hovsapian, "Resiliency-driven proactive distribution system reconfiguration with synchrophasor data," *IEEE Trans. Power Syst.*, vol. 35, no. 4, pp. 2748-2758, Jul. 2020.

[127] Z. Yang, H. Zhong, A. Bose, T. Zheng, Q. Xia, and C. Kang, "A linearized OPF model with reactive power and voltage magnitude: a pathway to improve the MW-only DC OPF," *IEEE Trans. Power Syst.*, vol. 33, no. 2, pp. 1734-1745, Mar. 2018.

[128] H. Yuan, F. Li, Y. Wei, and J. Zhu, "Novel linearized power flow and linearized OPF models for active distribution networks with application in distribution LMP," *IEEE Trans. Smart Grid*, vol. 9, no. 1, pp. 438-448, Jan. 2018.

[129] M. Li, A. Pal, A. G. Phadke, and J. S. Thorp, "Transient stability prediction based on apparent impedance trajectory recorded by PMUs," *Int. J. Elect. Power Energy Syst.*, vol. 54, pp. 498-504, Jan. 2014.

[130] C. Mishra, A. Pal, J. S. Thorp, and V. A. Centeno, "Transient stability assessment of prone-to-trip renewable generation rich power systems using Lyapunov's direct method," *IEEE Trans. Sustainable Energy*, vol. 10, no. 3, pp. 1523-1533, Jul. 2019.

[131] C. Mishra, R. Sen Biswas, A. Pal, and V. A. Centeno, "Critical clearing time sensitivity for inequality constrained systems," *IEEE Trans. Power Syst.*, vol. 35, no. 2, pp. 1572-1583, Mar. 2020.

[132] G. Hou, and V. Vittal, "Trajectory sensitivity based preventive control of voltage instability considering load uncertainties," *IEEE Trans. Power Syst.*, vol. 27, no. 4, pp. 2280-2288, Nov. 2012.

[133] S. Xia, Z. Ding, M. Shahidehpour, K. W. Chan, S. Bu and G. Li, "Transient stability-constrained optimal power flow calculation with extremely unstable conditions using energy sensitivity method," *IEEE Trans. Power Syst.*, vol. 36, no. 1, pp. 355-365, Jan. 2021.

[134] Y. Xu, Z. Y. Dong, R. Zhang, Y. Xue, and D. J. Hill, "A decomposition-based practical approach to transient stability-constrained unit commitment," *IEEE Trans. Power Syst.*, vol. 30, no. 3, pp. 1455-1464, May 2015.

[135] Y. Xu, J. Ma, Z. Y. Dong, and D. J. Hill, "Robust transient stability-constrained optimal power flow with uncertain dynamic loads," *IEEE Trans. Smart Grid*, vol. 8, no. 4, pp. 1911-1921, Jul. 2017.

[136] X. Zhao, H. n. Wei, J. Qi, P. Li, and X. Bai, "Frequency stability constrained optimal power flow incorporating differential algebraic equations of governor dynamics," *IEEE Trans. Power Syst.*, Sep. 2020.

[137] A. Pizano-Martinez, C. R. Fuerte-Esquivel, and D. Ruiz-Vega, "A new practical approach to transient stability-constrained optimal power flow," *IEEE Trans. Power Syst.*, vol. 26, no. 3, pp. 1686-1696, Aug. 2011.

[138] Y. Guo, K. Baker, E. Dall'Anese, Z. Hu, and T. H. Summers, "Data-based distributionally robust stochastic optimal power flow—part I: methodologies," *IEEE Trans. Power Syst.*, vol. 34, no. 2, pp. 1483-1492, Mar. 2019.

[139] X. Fang, B. M. Hodge, H. Jiang, and Y. Zhang, ''Decentralized wind uncertainty management: Alternating direction method of multipliers based distributionally-robust chance constrained optimal power flow,'' *Appl. Energy*, vol. 239, pp. 938–947, Apr. 2019.

[140] O. Mégel, J. L. Mathieu, and G. Andersson, "Hybrid stochastic-deterministic multiperiod DC optimal power flow," *IEEE Trans. Power Syst.*, vol. 32, no. 5, pp. 3934-3945, Sep. 2017.

[141] S. Xia, X. Luo, K. W. Chan, M. Zhou, and G. Li, "Probabilistic transient stability constrained optimal power flow for power systems with multiple correlated uncertain wind generations," *IEEE Trans. Sustain. Energy*, vol. 7, no. 3, pp. 1133-1144, Jul. 2016.

APPENDIX A

BRANCH REACTANCE DATA OF A SAMPLE 5-BUS SYSTEM

The branch reactance data for the 5-bus test system is provided below.

Branch	Branch Reactance (χ_{jk})
1-2	0.02
1-5	0.02
1-3	0.04
2-3	0.02
3-5	0.04
4-5	0.02
3-4	0.02

APPENDIX B

BRANCH REACTANCE DATA OF A SAMPLE 10-BUS SYSTEM

The branch reactance data for the 10-bus test system is provided below.

Branch	Branch Reactance (χ_{jk})
1-3	0.0476
1-4	0.0417
2-3	0.0476
2-9	0.0964
3-9	0.0864
4-6	0.0388
5-6	0.1727
5-10	0.0519
6-7	0.0230
7-8	0.0396
8-9	0.0216

APPENDIX C

DIFFERENT CASE-STUDIES ON THE IEEE 118-BUS TEST SYSTEM

Different case-studies involve different sequences of multiple successive outages on the IEEE 118-bus test system. The case-studies 1 through 35 are presented in this page.

S. No.	Outage 1	Outage 2	Outage 3	Outage 4	Outage 5	Outage 6
Case study 1	'15-33'	'19-34'	'38-37'	'42-49'	'49-66'	'66-67'
Case study 2	'23-24'	'22-23'	'26-30'	'32-113'		
Case study 3	'15-33'	'19-34'	'30-38'	'24-72'		
Case study 4	'105-106'	'106-107'	'100-104'			
Case study 5	'3-5'	'3-12'				
Case study 6	'5-11'	'4-5'	'11-12'			
Case study 7	'11-12'	'3-5'	'5-6'	'16-17'		
Case study 8	'15-19'	'17-18'	'19-34'			
Case study 9	'25-27'	'17-31'	'17-113'			
Case study 10	'30-17'	'33-37'	'8-5'	'23-25'	'25-27'	'19-34'
Case study 11	'17-31'	'31-32'				
Case study 12	'23-32'	'25-27'	'17-31'			
Case study 13	'19-34'	'34-37'	'35-37'			
Case study 14	'38-37'	'15-33'	'42-49'	'19-34'		
Case study 15	'30-38'	'33-37'	'19-34'			
Case study 16	'40-42'	'41-42'	'37-40'			
Case study 17	'42-49'	'37-40'				
Case study 18	'45-49'	'45-46'				
Case study 19	'49-51'	'53-54'				
Case study 20	'54-56'	'54-55'	'54-59'	'49-54'		
Case study 21	'56-59'	'54-56'	'49-50'	'55-56'		
Case study 22	'38-65'	'42-49'	'44-45'			
Case study 23	'64-65'	'62-66'	'66-67'	'49-54'	'49-50'	
Case study 24	'65-66'	'61-62'	'49-66'			
Case study 25	'65-68'	'49-69'	'47-69'			
Case study 26	'68-69'	'65-68'				
Case study 27	'69-70'	'70-75'	'74-75'			
Case study 28	'24-70'	'70-71'				
Case study 29	'70-75'	'69-75'	'70-74'	'75-77'		
Case study 30	'77-80'	'79-80'	'77-82'			
Case study 31	'81-80'	'69-77'	'75-77'			
Case study 32	'82-83'	'85-89'				
Case study 33	'85-89'	'85-88'				
Case study 34	'89-92'	'90-91'				
Case study 35	'92-94'	'92-93'	'92-100'	'92-102'		

The case-studies 36 through 41 are presented in this page.

S. No.	Outage 1	Outage 2	Outage 3	Outage 4	Outage 5	Outage 6
Case study 36	'80-96'	'80-97'	'94-96'	'94-95'		
Case study 37	'94-96'	'95-96'	'94-100'	'92-94'		
Case study 38	'94-100'	'92-100'	'92-102'	'80-98'		
Case study 39	'100-103'	'100-104'				
Case study 40	'103-105'	'104-105'	'100-106'			
Case study 41	'100-106'	'100-103'				

APPENDIX D

MATLAB PSEUDOCODE: THE FIRST COMPONENT

The MATLAB pseudocode of the proposed first component has been presented here. This is the main program which uses several user defined functions (presented in Appendix F) to implement different algorithms developed in this research.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
% Program Name: Main_program_first_component                                   %
%                                                                              %
% Program Description: This program implements the proposed first             %
% component (FT-RTCA-iCA) during successive outages in a                       %
% power network                                                                %
%                                                                              %
% Author: Reetam Sen Biswas                                                    %
% Arizona State University                                                     %
%                                                                              %
% Last Modified: 03/20/2020                                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc
clear all
close all
%% Load the input data:
mpc = loadcase('case118_J2.m'); % All data associated with the IEEE
% 118-bus system are loaded here.
% Transmission line ratings obtained from the surge impedance loading
% of transmission lines are used.
% Multiple circuit transmission lines are converted to an equivalent
% single circuit configuration.

load Data_118bus_J2.mat; % A subset of the data which are frequently
% used by the proposed analysis are stored in this data structure.
% This .mat file contains four matrices named 1) Bus, 2) Branch
% 3) Generator and 4) Load.
% (1) Bus: This matrix contains a single column with all bus numbers
%     in ascending order
% (2) Branch: The first and second column of this matrix contains the
%     "from bus" and "to bus" information for different branches.
%     The third, fourth and fifth columns contain branch resistance,
%     reactance, and susceptance respectively.
%     The sixth column contains active power flow through different
%     transmission lines in the base case.
%     The seventh column contains branch ratings.
%     The eight column contains branch statuses.
% (3) Generator: The first column contains the generator bus numbers.
%     The second column contains the power generation at
%     different generators of the system. The third and
%     fourth columns contain the maximum and minimum generation at
%     respective generators. The fifth and sixth columns contain the
%     linear and quadratic cost coefficients respectively.
% (4) Load: The first column contains the load bus numbers. The second
%     column contains the net power demand at a specific load bus. The
```

```
%     third column contains the cost of load shed. The fourth and fifth
%     columns contain the maximum and minimum power demands at specific
%     load buses respectively.

%% Initialize different matrices:
Generator(:,4) = zeros(length(Generator),1);
loc_negative = find(Generator(:,2)<0);
Generator(loc_negative,3) = Generator(loc_negative,2);
Generator(loc_negative,4) = Generator(loc_negative,2);
initial = 1;
continue_flag = 1;
K = [];
baseMVA = 100;
NoOfBus = length(Bus);
BusGraph = Bus;
BranchGraph = Branch(:,[1:2]);BranchGraph(:,3) =
Branch(:,7);BranchGraph(:,4) = Branch(:,8);
GeneratorGraph = Generator(:,[1:2]);
LoadGraph = Load(:,[1:2]);

%% Settings:
% Rank_limit controls the size of the contingency list
% used in RTCA
Rank_limit = 54;
% RoundOffFlag determines if the PTDF values will be
% approximated below a specified threshold.
% 0: no approximation; 1: approximation
RoundOffFlag = 0;

%% Build the "flow" and "latent capacity" graphs:
% The graphs are built based on the graph-theory based
% network flow algorithm
[ Flow, Capacity, A, ~ ] = NetworkFlowAlgo-
rithm(BusGraph,BranchGraph,GeneratorGraph,LoadGraph);

% An alternate way of building the "flow" and "latent capacity"
% graphs is to use a DC power flow solution in the base-case scenario
% Flow = sparse(NoOfBus,NoOfBus);
% Capacity = sparse(NoOfBus,NoOfBus);
% for i = 1:length(Branch(:,1))
%     Flow(Branch(i,1),Branch(i,2)) = Branch(i,6); % dc power flows
%     Flow(Branch(i,2),Branch(i,1)) = (-1)*Branch(i,6);
%     Capacity(Branch(i,1),Branch(i,2)) = Branch(i,7)-
Flow(Branch(i,1),Branch(i,2));
%     Capacity(Branch(i,2),Branch(i,1)) = Branch(i,7)-
Flow(Branch(i,2),Branch(i,1));
% end

%% Find the list of radial branches in the system:
[ Radial, ~ ] = FindRadial( Branch, A );

%% Create the PTDF, LODF, B and H matrices:
[ PTDF_true, PTDF, LODF, B_full, H_full, ~ ] = Cre-
ate_PTDF_LODF_B_H(Bus,Branch,RoundOffFlag);
```

```
%% Perform Contingency Ranking:
[Bompard_rank, ~] = ContingencyRanking(Bus, Branch, Load, Generator,
PTDF_true);

%% Perform RTCA using DC power flows using the results of contingency
ranking:
[ Vio, flag_vio_rtca, ~] =
DC_RTCA_Ranking(mpc,Bompard_rank,Rank_limit,Radial);
if (isempty(Vio)==0)
    Pre_Vio_L = Vio(:,1);
else
    Pre_Vio_L = [];
end

%% Perform feasibility test (FT) for all branches in the base-case sce-
nario:
[ CL_Sp_vio, CutsetStack_vio, EdgeList, flag_vio_ft, ~ ] = Feasibil-
ityTestBasecase( Flow, Capacity, A, BranchGraph );
K_rtca = [];
K_ft = [];
outage_number = 1;
Total_load_base = sum(Load(:,2));

while (continue_flag==1)
    if ((flag_vio_rtca==1) || (flag_vio_ft==1))
%% Display violations detected by RTCA:
        if (flag_vio_rtca==1)
            DisplayViolations_RTCA(Vio);
        end
%% Display violations detected by FT:
        if (flag_vio_ft==1)
            DisplayViolations_FT( CL_Sp_vio,CutsetStack_vio );
        end

%% Create inputs from RTCA for the iCA:
        if (isempty(Vio)==0)
            if ((initial==1) || (isempty(K_rtca)==1))
                K_rtca = Vio(:,1);
            else
                K_rtca = vertcat(K_rtca,K_rtca_new);
                K_rtca = unique(K_rtca);
            end
        end

%% Create inputs from FT for the iCA:
        if ((initial==1) || (isempty(K_ft)==1))
                [ K_ft, Tm, Cutset_FT, ~ ] = CreateInput_ODC(
CL_Sp_vio, CutsetStack_vio, Branch);
        else
                [ K_ft, Tm, Cutset_FT, ~] = Aug-
mentCutsetInfo(K_ft_new,Tm_new,Cutset_FT_new,K_ft,Tm,Cutset_FT);
        end
        count_unique = 1;
        K_ft_unique = [];
```

```
        Tm_unique = [];
        Cutset_FT_unique = [];
        [row_K_ft, ~] = size(K_ft);

        for i = 1:row_K_ft
            branch_num = K_ft(i);
            flag = IsPresent(K_rtca,branch_num);
            if (flag==0)
                K_ft_unique(count_unique,:) = K_ft(i,:);
                Tm_unique(count_unique,1) = Tm(i,1);
                Cutset_FT_unique(:,:,count_unique) = Cutset_FT(:,:,i);
                count_unique = count_unique + 1;
            end
        end

%% Perform the Integrated Corrective Action (iCA):
        [ GeneratorNegativeChange, GeneratorPositiveChange, LoadNega-
tiveChange, LoadPositiveChange, Branch, Load, Generator, Soln_Flag,
tot_change_cost, ~ ] = IntegratedCorrectiveAction(K_rtca, PTDF, LODF,
Bus, Branch, Generator, Load, Radial, K_ft_unique, Tm_unique,
Cutset_FT_unique);

%% Update the system based upon the redispatch solution:
        if (Soln_Flag==0)
            break;
        end
        mpc.gen(:,2) = Generator(:,2);
        for nload = 1:length(Load(:,1))
             LoadBus = Load(nload,1);
             loc = find(mpc.bus(:,1)==LoadBus);
             mpc.bus(loc,3) = Load(nload,2);
        end
        Res = rundcpf(mpc);
        Branch(:,6) = Res.branch(:,14);

%% Perform RTCA following redispatch:
        [ Vio, flag_vio_rtca, ~ ] = DC_RTCA_Ranking( mpc, Bompard_rank,
Rank_limit, Radial);
        if (isempty(Vio)==0)
            Vio_L = Vio(:,1);
            Intersect_Pre_Vio_L = intersect(Pre_Vio_L,Vio_L);
            if (length(Intersect_Pre_Vio_L)==length(Vio_L))
                flag_vio_rtca = 0;
            else
                Pre_Vio_L = Vio_L;
            end
        end
        if (flag_vio_rtca==1)
             K_rtca_new = Vio(:,1);
             initial = 0;
        end

        % Group all the injection increase and injection decrease
        % together in separate matrices
        InjectionPositiveChange = [];
```

```
        Temp = []; Temp = LoadNegativeChange; Temp(:,2) = (-
1)*Temp(:,2);
        InjectionPositiveChange = vertcat(GeneratorPosi-
tiveChange,Temp);
        Temp = [];Temp = LoadPositiveChange;
        InjectionNegativeChange = vertcat(GeneratorNega-
tiveChange,Temp);

 %% Update the "flow" and "latent capacity graphs" following redispatch
based on M-UPS algorithm:
        [Flow,Capacity,BranchFlowChange,~] = ModifiedUpdateScheme(Flow,
Capacity, InjectionPositiveChange, InjectionNegativeChange,
BranchGraph);

%% Perform shortlisting assets following redispatch based on M-SA algo-
rithm:
        [ShortlistedEdges, ~] = ModifiedShortlistAssets(Branch-
FlowChange, EdgeList, BranchGraph);

%% Perform Feasibility Test (FT) on shortlisted assets:
        [CL_Sp_vio, CutsetStack_vio, EdgeList, flag_vio_ft, ~] = Feasi-
bilityTestOnShortlist( Flow, Capacity, A, BranchGraph,
ShortlistedEdges, EdgeList );

        if (flag_vio_ft==1)
%% Create inputs for Optimal Dispatch Change (ODC) for new cutsets:
            [K_ft_new, Tm_new, Cutset_FT_new, ~ ] = CreateInput_cutset(
CL_Sp_vio, CutsetStack_vio, Branch);
        end
    else
%% Display the results of the corrective action:
         fprintf('-----------------------------------------------------
-----------\n');
         fprintf('The first component has alleviated all post-contin-
gency cut-set saturation and critical branch overloads \n');
         fprintf('-----------------------------------------------------
-----------\n');
         GeneratorCost_Ar = Generator(:,6).*Generator(:,2).^2+Genera-
tor(:,5).*Generator(:,2); TotalGeneratorCost = sum(GeneratorCost_Ar);
         LoadCost_Ar = (Load(:,2)-Load(:,4)).*Load(:,3); TotalLoadCost
= sum(LoadCost_Ar);
         fprintf('Production cost = $ %f \n',TotalGeneratorCost);
         Net_load_shed = Total_load_base-sum(Load(:,2));
         fprintf('Total amount of load shed = %f MW \n',Net_load_shed);

%% Check if there are successive branch outages in the system:
        LineOutNumber = input('\n Enter the branch number which is out
(Press 0 and enter if you do not want to continue) ?');
        fprintf('\n');
        fprintf('\n ******** New Outage: ************\n');

        if (LineOutNumber==0)
            continue_flag = 0;
            break;
        else
```

```
%% Update the system matrices following the branch outage:
            mpc.branch(LineOutNumber,11) = 0;
            Res = rundcpf(mpc);
            Branch(:,6) = Res.branch(:,14);
            Branch(:,8) = Res.branch(:,11);
            BranchGraph(:,4) = Branch(:,8);
            A(Branch(LineOutNumber,1),Branch(LineOutNumber,2)) = 0;
            A(Branch(LineOutNumber,2),Branch(LineOutNumber,1)) = 0;
% Update the system matrices, instead of re-building
% the matrices from scratch
            [ PTDF_true, PTDF, LODF, B_full, H_full, ~ ] = Up-
date_PTDF_LODF_B_H( B_full, H_full, Bus, Branch, LineOutNum-
ber,RoundOffFlag);

%% Find the radial branches for the new system:
            [ Radial, ~ ] = FindRadial( Branch, A );
%% Perform contingency ranking:
            [Bompard_rank, ~] = ContingencyRanking(Bus, Branch, Load,
Generator, PTDF_true);
%% Perform RTCA using DC power flows:
            [ Vio, flag_vio_rtca, ~ ] = DC_RTCA_Ranking( mpc,
Bompard_rank, Rank_limit, Radial);
             if (isempty(Vio)==0)
                    Vio_L = Vio(:,1);
                    Pre_Vio_L = Vio_L;
             end
             if (flag_vio_rtca==1)
                    K_rtca_new = Vio(:,1);
             end
%% A successive FT has to be performed following the branch outage
% However, the successive FT must involve the UPS algorithm and the
% SA algorithm for fast computation
            [ Flow, Capacity, A, CL_Sp_vio, EdgeList, PathStack, Edge-
SatStack, CutsetStack_vio, ~ ] = OutageAnalysis( BranchGraph, Flow, Ca-
pacity, LineOutNumber, EdgeList, A );
            [ row_vio, col_vio ] = size(CL_Sp_vio);
            flag_vio_ft = 0;
            if (row_vio>=1)
                flag_vio_ft = 1;
                initial = 0;
                [ K_ft_new, Tm_new, Cutset_FT_new, ~ ] = CreateIn-
put_cutset( CL_Sp_vio, CutsetStack_vio, Branch);
            end
            if ((flag_vio_rtca==0) && (flag_vio_ft==0))
                fprintf('\n There are no violations detected by the
Feasibility Test (FT) and DC-RTCA \n');
            end
            outage_number = outage_number + 1;
        end
    end
end
```

APPENDIX E

MATLAB PSEUDOCODE: THE SECOND COMPONENT

The MATLAB pseudocode of the proposed second component is presented here. This is the main program which uses several user defined functions (presented in Appendix F) to implement different algorithms developed in this research.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                             %
% Program Name: Main_program_second_component                                 %
%                                                                             %
% Program Description: This program implements the proposed second            %
% component (FT-rCA) during successive outages in a power network             %
%                                                                             %
% Author: Reetam Sen Biswas                                                   %
% Arizona State University                                                    %
%                                                                             %
% Last Modified: 03/20/2020                                                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%
clc
clear all
close all

%% Load the input data
%% Load the input data:

mpc = loadcase('case118_J2.m'); % All data associated with the IEEE
% 118-bus system are loaded here.
% Transmission line ratings obtained from the surge impedance loading
% of transmission lines are used.
% Multiple circuit transmission lines are converted to an equivalent
% single circuit configuration.

load Data_118bus_J2.mat; % A subset of the data which are frequently
% used by the proposed analysis are stored in this data structure.
% This .mat file contains four matrices named 1) Bus, 2) Branch
% 3) Generator and 4) Load.
% (1) Bus: This matrix contains a single column with all bus numbers
%     in ascending order
% (2) Branch: The first and second column of this matrix contains the
%     "from bus" and "to bus" information for different branches.
%     The third, fourth and fifth columns contain branch resistance,
%     reactance, and susceptance respectively.
%     The sixth column contains active power flow through different
%     transmission lines in the base case.
%     The seventh column contains branch ratings.
%     The eight column contains branch statuses.
% (3) Generator: The first column contains the generator bus numbers.
%     The second column contains the power generation at
%     different generators of the system. The third and
%     fourth columns contain the maximum and minimum generation at
```

```
%     respective generators. The fifth and sixth columns contain the
%     linear and quadratic cost coefficients.
% (4) Load: The first column contains the load bus numbers. The second
%     column contains the net power demand at a specific load bus. The
%     third column contains the cost of load shed. The fourth and fifth
%     columns contain the maximum and minimum power demands at specific
%     load buses respectively.

%% Initialize different matrices:
BusGraph = Bus;
BranchGraph = Branch(:,[1:2]);BranchGraph(:,3) =
Branch(:,7);BranchGraph(:,4) = Branch(:,8);
GeneratorGraph = Generator(:,[1:2]);
LoadGraph = Load(:,[1:2]);
GenBusNumAr = Generator(:,1);
PgenOldAr = Generator(:,2);

%% Build the "flow" and "latent capacity" graphs:
% The graphs are built based on the graph-theory based
% network flow algorithm
[ Flow, Capacity, A, ~ ] = NetworkFlowAlgo-
rithm(BusGraph,BranchGraph,GeneratorGraph,LoadGraph);

% An alternate way of building the "flow" and "latent capacity"
% graphs is to use a DC power flow solution in the base-case scenario
% Flow = sparse(NoOfBus,NoOfBus);
% Capacity = sparse(NoOfBus,NoOfBus);
% for i = 1:length(Branch(:,1))
%     Flow(Branch(i,1),Branch(i,2)) = Branch(i,6); % dc power flows
%     Flow(Branch(i,2),Branch(i,1)) = (-1)*Branch(i,6);
%     Capacity(Branch(i,1),Branch(i,2)) = Branch(i,7)-
Flow(Branch(i,1),Branch(i,2));
%     Capacity(Branch(i,2),Branch(i,1)) = Branch(i,7)-
Flow(Branch(i,2),Branch(i,1));
% end

%% Initialize different variables:
fprintf('\n------ System condition: Base-case (No outage) ------\n');
BranchOut = [];
count = 1;
flag_vio_out = 1;
Net_change_cost = 0;
continue_flag = 1;
K = [];
Cutset_FT = [];
Total_load_base = sum(Load(:,2));

%% Create the PTDF, LODF, B and H matrices:
fprintf('Creating the PTDF matrix \n');
baseMVA = 100;
NoOfBus = length(Bus);
RoundOffFlag = 0;
[PTDF_true, PTDF, LODF, B_full, H_full, ~] = Cre-
ate_PTDF_LODF_B_H(Bus,Branch,RoundOffFlag);
```

```
%% Perform feasibility test (FT) for all branches in the base-case sce-
nario:
[ CL_Sp_vio, CutsetStack_vio, EdgeList, flag_vio, ~ ] = Feasibil-
ityTestBasecase( Flow, Capacity, A, BranchGraph );

fprintf('---------------------------------------- \n');

initial = 1;

while (continue_flag==1)

if (flag_vio==1)
    %% Display the violations detected by the FT algorithm:
    DisplayViolations_FT( CL_Sp_vio,CutsetStack_vio );

    %% Create inputs for the relaxed corrective action (rCA):
    if ((initial==1) || (isempty(K)==1))
        [ K, Tm, Cutset_FT, ~ ] = CreateInput_cutset( CL_Sp_vio,
CutsetStack_vio, Branch);
    else
        [ K, Tm, Cutset_FT, ~] = Aug-
mentCutsetInfo(K_new,Tm_new,Cutset_FT_new,K,Tm,Cutset_FT);
    end

    while (flag_vio==1)
%% Perform the relaxed corrective action (rCA):
          [ GeneratorNegativeChange, GeneratorPositiveChange, LoadNeg-
ativeChange, LoadPositiveChange, Branch, Load, Generator, Soln_Flag,
Flow_dc, Rate_dc, tot_change_cost, ~ ] = RelaxedCorrectiveAction( K,
Tm, Cutset_FT, PTDF, Bus, Branch, Generator, Load );
          count = count + 1;
          % Group all the injection increase and injection decrease
          % together in separate matrices
          InjectionPositiveChange = [];
          Temp = []; Temp = LoadNegativeChange; Temp(:,2) = (-
1)*Temp(:,2);
          InjectionPositiveChange = vertcat(GeneratorPosi-
tiveChange,Temp);
          Temp = [];Temp = LoadPositiveChange;
          InjectionNegativeChange = vertcat(GeneratorNega-
tiveChange,Temp);
          if Soln_Flag==0
              % This implies that the optimization problem in the rCA
has
              % not converged and there is a problem
               break;
          end
%% Update the system based upon the redispatch solution:
          mpc.gen(:,2) = Generator(:,2);
          for nload = 1:length(Load(:,1))
               LoadBus = Load(nload,1);
               loc = find(mpc.bus(:,1)==LoadBus);
               mpc.bus(loc,3) = Load(nload,2);
          end
          Res = rundcpf(mpc);
```

```
            Branch(:,6) = Res.branch(:,14);

%% Update the "flow" and "latent capacity graphs" following redispatch
based on M-UPS:
            [Flow,Capacity,BranchFlowChange,~] = Modi-
fiedUpdateScheme(Flow, Capacity, InjectionPositiveChange, Injection-
NegativeChange, BranchGraph);

%% Perform shortlisting assets following redispatch based on M-SA algo-
rithm:
            [ShortlistedEdges, ~] = ModifiedShortlistAssets(Branch-
FlowChange, EdgeList, BranchGraph);

%% Perform Feasibility Test (FT) on shortlisted assets:
            [CL_Sp_vio, CutsetStack_vio, EdgeList, flag_vio, ~] = Feasi-
bilityTestOnShortlist( Flow, Capacity, A, BranchGraph,
ShortlistedEdges, EdgeList );

            if (flag_vio==1)
%% Display additional violations (if any) due to the redispatch:
                DisplayViolations_FT( CL_Sp_vio,CutsetStack_vio );
%% Create the inputs for rCA to mitigate the combined violations:
                [K_new, Tm_new, Cutset_FT_new, ~ ] = CreateInput_cutset(
CL_Sp_vio, CutsetStack_vio, Branch);
                [K, Tm, Cutset_FT, ~] = Aug-
mentCutsetInfo(K_new,Tm_new,Cutset_FT_new,K,Tm,Cutset_FT);
            end
    end
    if (flag_vio==0)
%% Display the results of the corrective action:
            fprintf('-----------------------------------------------------
-----------\n');
            fprintf('The second component has alleviated all post-contin-
gency cut-set saturation \n');
            fprintf('-----------------------------------------------------
-----------\n');
            GeneratorCost_Ar = Generator(:,6).*Generator(:,2).^2+Genera-
tor(:,5).*Generator(:,2); TotalGeneratorCost = sum(GeneratorCost_Ar);
            LoadCost_Ar = (Load(:,2)-Load(:,4)).*Load(:,3); TotalLoadCost
= sum(LoadCost_Ar);
            fprintf('Production cost = $ %f \n',TotalGeneratorCost);
            Net_load_shed = Total_load_base-sum(Load(:,2));
            fprintf('Total amount of load shed = %f MW \n',Net_load_shed);
    else
            fprintf('-----------------------------------------------------
-----------\n');
            fprintf('Warning: All post-contingency cut-set saturation can-
not be mitigated \n');
            fprintf('-----------------------------------------------------
-----------\n');
    end

else
    fprintf('-----------------------------------------------------------
-------\n');
```

```
    fprintf('No violations are detected by the FT algorithm \n');
    fprintf('-------------------------------------------------------
-------\n');
end

%% Check if there are successive branch outages in the system:
   LineOutNumber = input('\n Enter the branch number which is out
(Press 0 and enter if you do not want to continue) ?');
   if LineOutNumber==0
      continue_flag = 0;
      break;

   else
      fprintf('\n------ System condition: Outage of branch (%d-%d)-----
-\n',Branch(LineOutNumber,1),Branch(LineOutNumber,2));

%% Update the system matrices following the branch outage:
        mpc.branch(LineOutNumber,11) = 0;
% Update the system matrices, instead of re-building
% the matrices from scratch
        [ PTDF_true, PTDF, LODF, B_full, H_full, ~ ] = Up-
date_PTDF_LODF_B_H( B_full, H_full, Bus, Branch, LineOutNum-
ber,RoundOffFlag);
        Res = rundcpf(mpc);
        Branch(:,6) = Res.branch(:,14);
        Branch(:,8) = Res.branch(:,11);
        BranchGraph(:,4) = Branch(:,8);

%% A successive FT has to be performed following the branch outage
% However, the successive FT must involve the UPS algorithm and the
% SA algorithm for fast computation
      [ Flow, Capacity, A, CL_Sp_vio, EdgeList, PathStack, Edge-
SatStack, CutsetStack_vio, ~ ] = OutageAnalysis( BranchGraph, Flow, Ca-
pacity, LineOutNumber, EdgeList, A );
      [ row_vio, col_vio ] = size(CL_Sp_vio);
      if (row_vio>=1)
          flag_vio = 1;
          initial = 0;
% Creates input for the next relaxed corrective action (rCA):
          [ K_new, Tm_new, Cutset_FT_new, ~ ] = CreateInput_cutset(
CL_Sp_vio, CutsetStack_vio, Branch);
      else
          flag_vio = 0;
      end
   end
end
```

APPENDIX F

MATLAB PSEUDOCODE: USER DEFINED FUNCTIONS

The pseudocodes for different user defined functions to implement different algorithms developed in the course of this research are presented here. All the user defined functions are presented in alphabetical order of their names.

```
function [ K, Tm, Cutset_FT, time ] = Aug-
mentCutsetInfo(K_new,Tm_new,Cutset_FT_new,K,Tm,Cutset_FT)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                             %
% Program Description: The new violations identified by the FT, are           %
% augmented with the violations detected in a previous iteration,             %
% such that the corrective action can be initiated with respect               %
% to all the violations                                                       %
%                                                                             %
% Author: Reetam Sen Biswas                                                   %
% Arizona State University                                                    %
%                                                                             %
% Last Modified: 03/20/2020                                                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    tic;
    % Initialization:
    NumOfCritCutset = length(K_new(:,1));
    MaxNumBranchCritCutset = length(K_new(1,:));

    % Make the transfer margins of the saturated cut-sets
    % addressed in previous iteration as zero

    [ row_K_old, ~] = size(K);
    for i = 1:row_K_old
        Tm(i,1) = 0;
    end

    % Augment the new cutsets with their respective transfer margins
    ncutset = length(Tm)+1;
    [row_K, col_K] = size(K_new);
    for r = 1:length(K_new(:,1))
        K(ncutset,[1:col_K]) = K_new(r,[1:col_K]);
        Tm(ncutset,1) = Tm_new(r);
        [row_set, col_set] = size(Cutset_FT_new(:,:,r));
        Cutset_FT([1:row_set],[1:col_set],ncutset) =
Cutset_FT_new([1:row_set],[1:col_set],r);
        ncutset = ncutset+1;
    end

    time = toc;

end
```

```
function [ LoseFlag, PathAr, CurrentFlow, FlowCap, FlowInjAr, flag_Ra-
dial, EdgeSat, Cutset] = CheckIfLose_Cutset( LinesArray, Line, Flow,
Capacity, A )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                       %
% Program Description: This program evaluates if a specific            %
% transmission outage will create post-contingency cut-set              %
% saturation, based upon the FT algorithm                               %
%                                                                       %
% Author: Reetam Sen Biswas                                             %
% Arizona State University                                              %
%                                                                       %
% Last Modified: 03/20/2020                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    BusA = LinesArray(Line,1);
    BusB = LinesArray(Line,2);

    NewFlowSheet = Flow;
    NewFlowSheet(NewFlowSheet<0) = 0;

    [Bus1, Bus2, flow] = find(NewFlowSheet);
    found = 0;

% Obtain the current flow of the specified branch:
    for i = 1:length(Bus1)
        if ((Bus1(i)==BusA && Bus2(i)==BusB) || (Bus1(i)==BusB &&
Bus2(i)==BusA))
            FromBus = Bus1(i);
            ToBus = Bus2(i);
            CurrentFlow = flow(i);
            found = 1;
        end
    end
    if found==0
        FromBus = BusA;
        ToBus = BusB;
        CurrentFlow = 0;
    end

% Updates the incidence matrix accordingly:
    A(FromBus,ToBus) = 0;
    A(ToBus,FromBus) = 0;

% Checks if the outage branch is a radial branch or not
    [S,path]=graphshortestpath(A,FromBus,To-
Bus,'Method','BFS','Directed','true');
    flag_Radial = 0;
    if S==Inf
        flag_Radial = 1;
    end

% Remove the line from the latent capacity graph
    Capacity(FromBus,ToBus) = 0;
```

```
    Capacity(ToBus, FromBus) = 0;

%% Find the maximum power that can be transferred along the indirect
paths
    FlowCap = 0;
    LoseFlag = 0;
    FlowInjAr = [];
    countP = 1;
    PathAr = [];
    EdgeSat = [];
    countS = 1;
% If the power flow through the "direct path" is more than the "maximum
% power that can be transferred through the "indirect paths", then it
% creates a saturated cut-set.
    while (1<2)
      [S,path]=graphshortestpath(Capacity,FromBus,To-
Bus,'Method','BFS','Directed','true');
      if S==Inf
            break;
      end

      if S<Inf
            PathAr(countP,[1:length(path)]) = path;
            MaxCap = 9999;
            for k=1:S
                From = path(k);To = path(k+1);
                if MaxCap>Capacity(From,To)
                    MaxCap = Capacity(From,To);
                end
            end
            FlowInj = MaxCap;
            for k=1:S
                From = path(k);To = path(k+1);
                Flow(From,To) = Flow(From,To) + FlowInj;
                Flow(To,From) = Flow(To,From) - FlowInj;
                Capacity(From,To) = Capacity(From, To) - FlowInj;
                Capacity(To,From) = Capacity(To, From) + FlowInj;
                if Capacity(From,To)<0.0001
                    % Finding the saturated edges after flow injection:
                    EdgeSat(countS,1) = From;
                    EdgeSat(countS,2) = To;
                    countS = countS + 1;
                end
            end
            FlowCap = FlowCap + FlowInj;
            FlowInjAr(countP,1) = FlowInj;
            countP = countP + 1;
            if FlowCap>=CurrentFlow
                LoseFlag = 1;
                break;
            end
      end
    end
% Saturated cut-sets with a transfer margin lesser than 0.001 are ig-
nored
```

```
    if (abs(CurrentFlow-FlowCap)<0.001)
        LoseFlag = 1;
    end

    Cutset = [];

%% Find the saturated cut-set:
    if LoseFlag==0
        V_insub = [];
        if LoseFlag==0 && flag_Radial==0
            % Group the vertices:
            [row_P, col_P] = size(PathAr);
            for i = 1:row_P
                V_insub = horzcat(V_insub,PathAr(i,:));
            end
            V_insub = unique(V_insub);
            V_insub(V_insub==0) = [];
        end

        V_reach_F = []; V_reach_T = [];
        countF = 1; countT = 1;
        [row_V, col_V] = size(V_insub);
        for v = 1:col_V
            [S,path]=graphshortestpath(Capacity,FromBus,V_in-
sub(v),'Method','BFS','Directed','true');
            if S<Inf
                V_reach_F(countF,1) = V_insub(v); countF = countF+1;
            else
                V_reach_T(countT,1) = V_insub(v); countT = countT+1;
            end
        end

        K = 1;
        Cutset(K,1) = FromBus;Cutset(K,2) = ToBus; K = K+1;
        [row_E, col_E] = size(EdgeSat);

        for i = 1:row_E
            F = EdgeSat(i,1); T = EdgeSat(i,2);
            [ flag_F, pos ] = IsPresent( V_reach_F, F );
            [ flag_T, pos ] = IsPresent( V_reach_T, T );
            if flag_F==1 && flag_T==1
                Cutset(K,1) = F;
                Cutset(K,2) = T;
                K = K+1;
            end
        end
    end

end
```

```
function [ T_sort, timeBomp ] = ContingencyRanking( Bus, Branch, Load,
Generator, PTDF )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                       %
% Program Description: This program finds the contingency ranking      %
% based upon the PTDFs and branch ratings                               %
%                                                                       %
% Author: Reetam Sen Biswas                                             %
% Arizona State University                                              %
%                                                                       %
% Last Modified: 03/20/2020                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

tic;
NoOfGen = length(Generator(:,1));
NoOfLoad = length(Load(:,1));
NoOfBranch = length(Branch(:,1));
NoOfBus = length(Bus(:,1));
C_gd = zeros(NoOfGen,NoOfLoad);
Pl_max = Branch(:,7);
zero_col = zeros(NoOfBranch,1);
PTDF = horzcat(PTDF,zero_col);

for g = 1:NoOfGen
    gbus = Generator(g,1);
    for d = 1:NoOfLoad
        dbus = Load(d,1);
        if (gbus~=dbus)

            if (gbus~=NoOfBus)
                ptdf_lines_gbus = PTDF(:,gbus);
            else
                ptdf_lines_gbus = zeros(NoOfBranch,1);
            end

            if (dbus~=NoOfBus)
                ptdf_lines_dbus = PTDF(:,dbus);
            else
                ptdf_lines_dbus = zeros(NoOfBranch,1);
            end

            ptdf_lines_gbus_dbus = ptdf_lines_gbus - ptdf_lines_dbus;

            value_ar = Pl_max./abs(ptdf_lines_gbus_dbus);

            value = min(value_ar);

            C_gd(g,d) = value;

        end
    end
end

for nline = 1:NoOfBranch
```

```
        gen_buses = Generator(:,1);
        load_buses = Load(:,1);

        ptdf_gen = PTDF(nline,gen_buses)';
        ptdf_load = PTDF(nline,load_buses);

        ptdf_gen_mat = repmat(ptdf_gen,[1,NoOfLoad]);

        ptdf_load_mat = repmat(ptdf_load,[NoOfGen,1]);

        ptdf_gen_load_mat = ptdf_gen_mat - ptdf_load_mat;

        common = intersect(gen_buses,load_buses');

        for j = 1:length(common)
           gen_loc = find(gen_buses==common(j));
           load_loc = find(load_buses==common(j));
           ptdf_gen_load_mat(gen_loc,load_loc) = 0;
        end

        [r,c] = find(ptdf_gen_load_mat<0);
        ptdf_gen_load_mat_p = ptdf_gen_load_mat;

        for k = 1:length(r)
            ptdf_gen_load_mat_p(r(k),c(k)) = 0;
        end

        ptdf_weight_positive = ptdf_gen_load_mat_p.*C_gd;
        Tp = sum(sum(ptdf_weight_positive));

        [r,c] = find(ptdf_gen_load_mat>0);
        ptdf_gen_load_mat_n = ptdf_gen_load_mat;

        for k = 1:length(r)
            ptdf_gen_load_mat_n(r(k),c(k)) = 0;
        end

        ptdf_weight_negative = ptdf_gen_load_mat_n.*C_gd;
        Tn = sum(sum(ptdf_weight_negative));

      T(nline,1) = nline;
      T(nline,2) = Branch(nline,1);
      T(nline,3) = Branch(nline,2);
      T(nline,4) = max(Tp,abs(Tn));

end
T_max = max(T(:,4));
T(:,4) = T(:,4)./T_max;
T_sort = sortrows(T,4,'descend');
timeBomp = toc;

end
```

```
function [ PTDF_true, PTDF_approx, LODF, B_full, H_full, time ] = Cre-
ate_PTDF_LODF_B_H(Bus,Branch,RoundOffFlag)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                       %
% Program Description: This function creates the power transfer.       %
% distribution factor (PTDF), line outage distribution factor (LODF),   %
% the susceptance matrix (B) and the branch-bus matrix (H)              %
%                                                                       %
% Author: Reetam Sen Biswas                                             %
% Arizona State University                                              %
%                                                                       %
% Last Modified: 03/20/2020                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    tic;
    %% Create the H (branch-bus matrix):
    H = [];
    for i = 1:length(Branch(:,1))
        Status = Branch(i,8);
        if Status==1
           FromBus = Branch(i,1);
           ToBus = Branch(i,2);
           H(i,FromBus) = 1/Branch(i,4);
           H(i,ToBus) = (-1)*1/Branch(i,4);
        else
            FromBus = Branch(i,1);
            ToBus = Branch(i,2);
            H(i,FromBus) = 0;
            H(i,ToBus) = 0;
        end
    end

    %% Build the B matrix after monitoring the branch statuses:
    B = zeros(length(Bus(:,1)));
    BusOld = Bus(:,1);
    for i = 1:length(Branch(:,1))
        Status = Branch(i,8);
        if Status==1
            FromBus = Branch(i,1);
            ToBus = Branch(i,2);
            xline = Branch(i,4);
            B(FromBus,ToBus) = B(FromBus,ToBus)-1/(xline);
            B(ToBus,FromBus) = B(ToBus,FromBus)-1/(xline);
            B(FromBus,FromBus) = B(FromBus,FromBus)+1/(xline);
            B(ToBus,ToBus) = B(ToBus,ToBus)+1/(xline);
        end
    end

    %% Adjust the B and H matrices to account for the reference bus
    % B matrix: Remove the entire row and column for the reference bus
    % H matrix: Remove the reference bus column from the H matrix
    noofbus = length(B(:,1));
    B_full = B;
```

```
    H_full = H;
    B_temp = B([1:noofbus-1],[1:noofbus-1]);
    B = B_temp;
    H_temp = H(:,1:noofbus-1);
    H = H_temp;

   %% Perform matrix operation to obtain the PTDF matrix
    X = inv(B);
    PTDF = H*X;
    PTDF_true = PTDF;

    if (RoundOffFlag==1)
        [r,c] = find(abs(PTDF)<0.02);
        for i = 1:length(r)
            PTDF(r(i),c(i)) = 0;
        end
    end
    PTDF_approx = PTDF;

    %% From the PTDF matrix, we now create the LODF matrix
    PTDF_full = horzcat(PTDF,zeros(length(Branch(:,1)),1));
    [nl, nb] = size(PTDF_full);
    f = Branch(:, 1);
    t = Branch(:, 2);
    Cft =  sparse([f; t], [1:nl 1:nl]', [ones(nl, 1); -ones(nl, 1)],
nb, nl);
    H = PTDF_full * Cft;
    h = diag(H, 0);
    LODF = H ./ (ones(nl, nl) - ones(nl, 1) * h');
    h_diff = abs(ones(length(h),1)-h);
    [ pos_ar ] = find(h_diff<0.00001);
    LODF = LODF - diag(diag(LODF)) - eye(nl, nl);
    for  i = 1:length(pos_ar)
       pos_val = pos_ar(i);
       LODF([1:nl],pos_val) = zeros(nl,1);
       LODF(pos_val,[1:nl]) = zeros(1,nl);
       LODF(pos_val,pos_val) = -1;
    end
    time = toc;
end
```

```
function [ K, Tm, Cutset_FT, time ] = CreateInput_cutset( CL_Sp_vio,
CutsetStack_vio,Branch)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                             %
% Program Description: This program creates the inputs from                   %
% the FT algorithm to be utilized in the corrective actions:                  %
% either iCA or rCA                                                           %
%                                                                             %
% Author: Reetam Sen Biswas                                                   %
% Arizona State University                                                    %
%                                                                             %
% Last Modified: 03/20/2020                                                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    tic;
    [noofcutset,col] = size(CL_Sp_vio);
    [xdim,ydim,zdim] = size(CutsetStack_vio);
    ncutset = 1;
    K = []; Tm = 0; Cutset_FT = [];
    for r = 1:noofcutset
        Tm(ncutset,1) = CL_Sp_vio(r,4);
        for i = 1:xdim
            F = CutsetStack_vio(i,1,r);
            T = CutsetStack_vio(i,2,r);
            if F~=0
               Cutset_FT(i,1,ncutset) = F;Cutset_FT(i,2,ncutset) = T;
               for j = 1:length(Branch(:,1))
                   if (F==Branch(j,1) && T==Branch(j,2)) ||
(F==Branch(j,2) && T==Branch(j,1))
                       K(ncutset,i) = j;
                       break;
                   end
               end
            end
        end
        ncutset = ncutset + 1;
    end
    time = toc;

end
```

```
function [ out ] = DisplayViolations_FT( CL_Sp_vio,CutsetStack_vio )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                             %
% Program Description: This program displays the violations                   %
% (post-contingency cut-set saturation) identified by the FT                  %
% algorithm                                                                   %
%                                                                             %
% Author: Reetam Sen Biswas                                                   %
% Arizona State University                                                    %
%                                                                             %
% Last Modified: 03/20/2020                                                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[ row_Cl,~ ] = size(CL_Sp_vio);
if row_Cl>0
    fprintf('------------------------------------------ \n');
    fprintf('Contingencies that create saturated cut-sets: \n');
    fprintf('------------------------------------------ \n');
    for i = 1:length(CL_Sp_vio(:,1))
        fprintf('Case %d :',i);
        fprintf('Outage of %d-%d saturates cut-set K%d by %f MW, where
K%d={',CL_Sp_vio(i,2),CL_Sp_vio(i,3),i,CL_Sp_vio(i,4),i);
        [row,col] = size(CutsetStack_vio(:,:,1));
        for j = 1:row
            F = CutsetStack_vio(j,1,i);
            T = CutsetStack_vio(j,2,i);
            LastFlag = 0;
            if j==row
                LastFlag = 1;
            else
                if CutsetStack_vio(j+1,1,i)==0
                    LastFlag = 1;
                end
            end

            if F>0
                if (LastFlag==0)
                    fprintf('%d-%d,',F,T);
                else
                    fprintf('%d-%d',F,T);
                end
            end
        end
        fprintf('} \n');
    end
    fprintf('------------------------------------------ \n');
else
   fprintf('------------------------------------------ \n');
   fprintf('No contingencies create saturated cut-sets: \n');
   fprintf('------------------------------------------ \n');
end
out = 1;
end
```

```
function [out] = DisplayViolations_RTCA(CL_Sp_vio)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                             %
% Program Description: This program displays the violations                   %
% detected by the RTCA. RTCA identifies critical contingencies                %
% that create post-contingency branch overloads.                              %
%                                                                             %
% Author: Reetam Sen Biswas                                                   %
% Arizona State University                                                    %
%                                                                             %
% Last Modified: 03/20/2020                                                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    fprintf('------------------------------------------- \n');
    fprintf('Contingencies that create post-contingency branch over-
loads are as follows: \n');
    fprintf('------------------------------------------- \n');
    [row, ~] = size(CL_Sp_vio);
    for i = 1:row
       fprintf('%f-%f \n',CL_Sp_vio(i,2),CL_Sp_vio(i,3));
    end
    out = 1;
    fprintf('------------------------------------------- \n');
end
```

```
function [CL_Sp_vio, CutsetStack_vio, EdgeList, flag_vio, time] = Fea-
sibilityTestBasecase( Flow, Capacity, A, Branch )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                             %
% Program Description: This program implements the feasibility               %
% test (FT) algorithm in the base-case scenario for all                       %
% transmission assets                                                         %
%                                                                             %
% Author: Reetam Sen Biswas                                                   %
% Arizona State University                                                    %
%                                                                             %
% Last Modified: 03/20/2020                                                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

tic;
% Find the lines which have a latent capacity of zero,
% and increase its capacity by a small margin
for Line=1:length(Branch(:,1))
    if Capacity(Branch(Line,1),Branch(Line,2))==0
        Capacity(Branch(Line,1),Branch(Line,2)) = 0.0001;
    elseif Capacity(Branch(Line,2),Branch(Line,1))==0
        Capacity(Branch(Line,2),Branch(Line,1)) = 0.0001;
    end
end

%% Analyzing different transmission assets by the FT algorithm:
CL_Sp = [];
count = 1;
CL_Sp_vio = [];
CutsetStack_vio = [];
EdgeList = zeros(length(Branch(:,1)),1);
count_radial = 1;
for Line=1:length(Branch(:,1))
        FlagPresBefore = 0;
        if FlagPresBefore==0
            [ LoseFlag, PathAr, CurrentFlow, FlowCap, FlowInjAr,
flag_Radial, EdgeSat, Cutset ] = CheckIfLose_Cutset(Branch, Line, Flow,
Capacity, A);
            [ row, col ] = size(PathAr);
            EdgeCount = 1;
            EnterLoop = 0;
            for R = 1:row
                for C = 1:col-1
                    if PathAr(R,C+1)>0
                        PresentFlag = 0;
                        if EnterLoop==1
                            Col_list = length(EdgeList(Line,:));
                            for k = 1:Col_list-1
                                if EdgeList(Line,k)==PathAr(R,C) &&
EdgeList(Line,k+1)==PathAr(R,C+1)
                                    PresentFlag=1;
                                end
                            end
                        end
```

```
                        if PresentFlag==0
                            EdgeList(Line,EdgeCount) = PathAr(R,C);
                            EdgeList(Line,EdgeCount+1) = PathAr(R,C+1);
                            EdgeCount = EdgeCount+2;
                        end
                        EnterLoop = 1;
                    end
                end
            end

            if LoseFlag==0
                CL_Sp(count,1) = Line;
                CL_Sp(count,2) = Branch(Line,1);
                CL_Sp(count,3) = Branch(Line,2);
                CL_Sp(count,4) = FlowCap-CurrentFlow;
                CL_Sp(count,5) = flag_Radial;
                [row,col] = size(PathAr);
                PathStack([1:row],[1:col],count) = PathAr;
                [row_e,col_e] = size(EdgeSat);
                EdgeSatStack([1:row_e],[1:col_e],count) = EdgeSat;
                [row_K,col_K] = size(Cutset);
                CutsetStack([1:row_K],[1:col_K],count) = Cutset;
                count = count + 1;
            end
            NoOfPaths(Line,1) = size(PathAr,1);
        end
end

%% Check if there are non-radial special assets detected by the FT al-
gorithm
flag_vio = 0;
count = 1;

if (isempty(CL_Sp)==1)
    CL_Sp_vio = [];
    CutsetStack_vio = [];
else
    for i = 1:length(CL_Sp(:,1))
        value = CL_Sp(i,5);
        if value==0
            flag_vio = 1;
            CL_Sp_vio(count,:) = CL_Sp(i,:);
            [r,c] = size(CutsetStack(:,:,i));
            CutsetStack_vio([1:r],[1:c],count) =
CutsetStack([1:r],[1:c],i);
            count = count + 1;
        end
    end
end

fprintf('\n Total number of paths traversed = %d \n',sum(NoOfPaths));
num_non_zero = length(find(NoOfPaths~=0));
fprintf('Average number of paths traversed = %d
\n',sum(NoOfPaths)/num_non_zero);
```

```
time = toc;
end
```

```
function [ CL_Sp_vio, CutsetStack_vio, EdgeList, flag_vio, time ] =
FeasibilityTestOnShortlist( Flow, Capacity, A, Branch,
ShortlistedEdges, EdgeList )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                             %
% Program Description: This program performs the feasibility test     %
% (FT) algorithm on the shortlisted assets following a change in       %
% generation redispatch in the system                                  %
%                                                                             %
% Author: Reetam Sen Biswas                                           %
% Arizona State University                                            %
%                                                                             %
% Last Modified: 03/20/2020                                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Initialization:
CL_Sp = [];
PathStack = [];
EdgeSatStack = [];
CutsetStack = [];
tic;
count = 1;
[rowF, colF] = size(ShortlistedEdges);

% Evaluate the shortlisted branches by the FT algorithm:
for i=1:rowF
    Line = ShortlistedEdges(i,1);
    [ LoseFlag, PathAr, CurrentFlow, FlowCap, FlowInjAr, flag_Radial,
EdgeSat, Cutset ] = CheckIfLose3_Break_Cutset( Branch, Line, Flow, Ca-
pacity, A );
    [ row, col ] = size(PathAr);
    EdgeCount = 1;
    EnterLoop = 0;
    EdgeList(Line,:) = zeros(1,length(EdgeList(Line,:)));
    for R = 1:row
        for C = 1:col-1
            if PathAr(R,C+1)>0
                PresentFlag = 0;
                if EnterLoop==1
                    Col_list = length(EdgeList(Line,:));
                    for k = 1:Col_list-1
                        if EdgeList(Line,k)==PathAr(R,C) && Edge-
List(Line,k+1)==PathAr(R,C+1)
                            PresentFlag=1;
                        end
                    end
                end

                if PresentFlag==0
                    EdgeList(Line,EdgeCount) = PathAr(R,C);
                    EdgeList(Line,EdgeCount+1) = PathAr(R,C+1);
                    EdgeCount = EdgeCount+2;
                end
                EnterLoop = 1;
            end
```

```
        end
    end
    [row, col] = size(PathAr);
    [row_e, col_e] = size(EdgeSat);
    [row_K,col_K] = size(Cutset);
    if LoseFlag==0
        PathInterest([1:row],[1:col],count) = PathAr;
        CL_Sp(count,1) = Line;
        CL_Sp(count,2) = Branch(Line,1);
        CL_Sp(count,3) = Branch(Line,2);
        CL_Sp(count,4) = FlowCap-CurrentFlow;
        CL_Sp(count,5) = flag_Radial;
        PathStack(1:row,1:col,count) = PathAr;
        EdgeSatStack(1:row_e,1:col_e,count) = EdgeSat;
        CutsetStack([1:row_K],[1:col_K],count) = Cutset;
        count = count + 1;
    end
end

% Check if there are non-singleton violations:
flag_vio = 0;
count = 1;
CL_Sp_vio = [];
CutsetStack_vio = [];
[row,col] = size(CL_Sp);
for i = 1:row
    value = CL_Sp(i,5);
    if value==0
        flag_vio = 1;
        CL_Sp_vio(count,:) = CL_Sp(i,:);
        [r,c] = size(CutsetStack(:,:,i));
        CutsetStack_vio([1:r],[1:c],count) =
CutsetStack([1:r],[1:c],i);
        count = count + 1;
    end
end
time = toc;

end
```

```
function [ Radial, time ] = FindRadial( Branch, A )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
% Program Description: This program finds the list of radial                   %
% branches in the system                                                       %
%                                                                              %
% Author: Reetam Sen Biswas                                                    %
% Arizona State University                                                     %
%                                                                              %
% Last Modified: 03/20/2020                                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

tic;
    Radial = [];
    count = 1;
    for i = 1:length(Branch(:,1))
        Fbus = Branch(i,1);
        Tbus = Branch(i,2);
        if (Branch(i,8)==1)
            A(Fbus,Tbus) = 0;
            A(Tbus,Fbus) = 0;
            [S,path]=graphshort-
estpath(A,Fbus,Tbus,'Method','BFS','Directed','true');
            if (S==Inf)
                Radial(count,1) = i;
                Radial(count,2) = Fbus;
                Radial(count,3) = Tbus;
                count = count + 1;
            end
            A(Fbus,Tbus) = 1;
            A(Tbus,Fbus) = 1;
        end
    end
time = toc;

end
```

```
function [ flag ] = IfCloseToZero( num )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                             %
% Program Description: This function checks if a number is                    %
% close to zero                                                               %
%                                                                             %
% Author: Reetam Sen Biswas                                                   %
% Arizona State University                                                    %
%                                                                             %
% Last Modified: 03/20/2020                                                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    if num<10^-4 && num>(-1)*10^4
        flag = 1;
    else
        flag = 0;
    end

end
```

```
function [ GeneratorNegativeChange, GeneratorPositiveChange, LoadNega-
tiveChange, LoadPositiveChange, Branch, Load, Generator, Soln_Flag,
tot_change_cost, time ] = IntegratedCorrectiveAction(K_rtca, PTDF,
LODF, Bus, Branch, Generator, Load, RadialLines,K_ft_unique, Tm,
Cutset_FT )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                             %
% Program Description: This program solves the optimization                   %
% problem for the integrated corrective action (iCA),                         %
% used in the proposed first component                                        %
%                                                                             %
% Author: Reetam Sen Biswas                                                   %
% Arizona State University                                                    %
%                                                                             %
% Last Modified: 03/20/2020                                                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tic;

%% Set-up the objective function:
    b = Generator(:,5);
    c = Generator(:,6);
    Pg_old_ar = Generator(:,2);
    f_gen_lin = (2*(Pg_old_ar.*c) + b);
    f_load = Load(:,3);
    f = vertcat(f_gen_lin,f_load);

    noofline = length(Branch(:,1));
    noofgen = length(Generator(:,1));
    noofload = length(Load(:,1));

    row_K = size(K_rtca,1);
    ContingencySet = [];
    count = 1;
    for lnum = 1:row_K
        if (IsPresent(RadialLines,K_rtca(lnum,1))~=1) &&
(Branch(K_rtca(lnum,1),8)==1)
            ContingencySet(count,1) = K_rtca(lnum,1);
            count = count+1;
        end
    end
    noofconting = size(ContingencySet,1);

%% Constraints for the conservation of energy:
    Pivot = 1;
    count_Sa = 1;
    for i = 1:noofgen
       Xa(count_Sa,1) = Pivot;Ya(count_Sa,1) = i;Va(count_Sa,1) = 1;
       count_Sa = count_Sa + 1;
    end
    for i = noofgen+1:(noofgen+noofload)
       Xa(count_Sa,1) = Pivot;Ya(count_Sa,1) = i;Va(count_Sa,1) = -1;
       count_Sa = count_Sa + 1;
    end
    Rhs_conserve = [0];
```

```
    Sign_conserve = [ '=' ];

    %% Constraints for the power injection limits:
    % Constraints for the injection maximum limit:
    Constraint_pinj = eye(noofgen+noofload,noofgen+noofload);
    count_Sb = 1;
    for i = 1:(noofgen+noofload)
         Xb(count_Sb,1) = Pivot+i;Yb(count_Sb,1) = i;Vb(count_Sb,1) =
1;
        count_Sb = count_Sb+1;
    end
    Pivot = Pivot+noofgen+noofload;
    % Constraints for the injection minimum limit:
    X_val = Pivot+1:Pivot+noofgen+noofload; X_val = X_val';
    Xb = vertcat(Xb,X_val);
    Yb = repmat(Yb,2,1);
    Vb = repmat(Vb,2,1);
    Pivot = Pivot+noofgen+noofload;
    for ngen = 1:noofgen
        GenBusNum = Generator(ngen,1);
        Pgen_old = Generator(ngen,2);
        Pgen_max = Generator(ngen,3);
        Pgen_min = Generator(ngen,4);
        Rhs_pinj_max(ngen,1) = Pgen_max-Pgen_old;
        Rhs_pinj_min(ngen,1) = Pgen_min-Pgen_old;
        Sign_pinj_max(ngen,1) = '<';
        Sign_pinj_min(ngen,1) = '>';
    end

    %% LHS and RHS for the injection limits:
    for nload = 1:noofload
        LoadBusNum = Load(nload,1);
        Pload_old = Load(nload,2);
        Rhs_pinj_max(noofgen+nload,1) = 0;
        Rhs_pinj_min(noofgen+nload,1) = -Pload_old;
        Sign_pinj_max(noofgen+nload,1) = '<';
        Sign_pinj_min(noofgen+nload,1) = '>';
    end

%% Constraints for pre-contingency power flow in each branch:
    Constraint_flow = zeros(noofline,noofgen+noofload);
    count_Sc = 1;
    noofflow_cstr = 0;
    for nline = 1:noofline
        flag_flow = 0;
        if (Branch(nline,8)==1)
            for ngen = 1:noofgen
                GenBusNum = Generator(ngen,1);
                if (GenBusNum==length(Bus))
                    PTDF_val = 0;
                else
                    PTDF_val = PTDF(nline,GenBusNum);
                end
                if (PTDF_val~=0)
                    Constraint_flow(nline,ngen) = PTDF_val;
```

```
                    Xc(count_Sc,1) = Pivot+noofflow_cstr+1;
Yc(count_Sc,1) = ngen; Vc(count_Sc,1) = PTDF_val;
                    count_Sc = count_Sc+1;
                    flag_flow = 1;
                end
            end
            for nload = 1:noofload
                LoadBusNum = Load(nload,1);
                if (LoadBusNum==length(Bus))
                    PTDF_val = 0;
                else
                    PTDF_val = PTDF(nline,LoadBusNum);
                end
                if (PTDF_val~=0)
                    Constraint_flow(nline,noofgen+nload) = (-
1)*PTDF_val;
                    Xc(count_Sc,1) = Pivot+noofflow_cstr+1;
Yc(count_Sc,1) = noofgen+nload; Vc(count_Sc,1) = (-1)*PTDF_val;
                    count_Sc = count_Sc+1;
                    flag_flow = 1;
                end
            end
            if (flag_flow==1)
                flow_old = Branch(nline,6);
                flow_max = Branch(nline,7);
                flow_min = (-1)*Branch(nline,7);
                Rhs_MaxFlow(noofflow_cstr+1,1) = flow_max-flow_old;
                Rhs_MinFlow(noofflow_cstr+1,1) = flow_min-flow_old;
                Sign_Maxflow(noofflow_cstr+1,1) = '<';
                Sign_Minflow(noofflow_cstr+1,1) = '>';
                noofflow_cstr = noofflow_cstr + 1;
            end
        end
    end
    Pivot = Pivot + noofflow_cstr;
    X_val = Xc+noofflow_cstr*ones(length(Xc),1);
    Xc = vertcat(Xc,X_val);
    Yc = repmat(Yc,2,1);
    Vc = repmat(Vc,2,1);
    Pivot = Pivot + noofflow_cstr;

%% Constraints for post-contingency branch flows:
    count_post = 1;
    count_Sd = 1;
    noofpostconting_cstr = 0;
    Xd = []; Yd = []; Vd = [];Rhs_MaxFlow_post = [];Rhs_MinFlow_post =
[];
    Sign_Maxflow_post = [];Sign_Minflow_post = [];

    for Cline = 1:size(ContingencySet,1)
        l = ContingencySet(Cline,1);
        flow_old_l = Branch(l,6);
        for k = 1:noofline
            flag_flow_post = 0;
             if (Branch(k,8)==1)
```

```
                    LODF_k_l = LODF(k,l);
                    for ngen = 1:noofgen
                        GenBusNum = Generator(ngen,1);
                        if (GenBusNum==length(Bus))
                            PTDF_k = 0;
                            PTDF_l = 0;
                        else
                            PTDF_k = PTDF(k,GenBusNum);
                            PTDF_l = PTDF(l,GenBusNum);
                        end
                        Value = PTDF_k+PTDF_l*LODF_k_l;
                        if (Value~=0)
                            Xd(count_Sd,1) = Pivot+count_post;
Yd(count_Sd,1) = ngen; Vd(count_Sd,1) = Value;
                            count_Sd = count_Sd + 1;
                            flag_flow_post = 1;
                        end
                    end
                    for nload = 1:noofload
                        LoadBusNum = Load(nload,1);
                        if (LoadBusNum==length(Bus))
                            PTDF_k = 0;
                            PTDF_l = 0;
                        else
                            PTDF_k = PTDF(k,LoadBusNum);
                            PTDF_l = PTDF(l,LoadBusNum);
                        end
                        Value = (-1)*PTDF_k + (-1)*PTDF_l*LODF_k_l;
                        if (Value~=0)
                            Xd(count_Sd,1) = Pivot+count_post;
Yd(count_Sd,1) = noofgen+nload; Vd(count_Sd,1) = Value;
                            count_Sd = count_Sd + 1;
                            flag_flow_post = 1;
                        end
                    end
                    if (flag_flow_post==1)
                        flow_old_k = Branch(k,6);
                        flow_max_k = Branch(k,7);
                        flow_min_k = (-1)*Branch(k,7);
                        Rhs_MaxFlow_post(noofpostconting_cstr+1,1) =
flow_max_k-(flow_old_k+flow_old_l*LODF_k_l);
                        Rhs_MinFlow_post(noofpostconting_cstr+1,1) =
flow_min_k-(flow_old_k+flow_old_l*LODF_k_l);
                        Sign_Maxflow_post(noofpostconting_cstr+1,1) = '<';
                        Sign_Minflow_post(noofpostconting_cstr+1,1) = '>';
                        noofpostconting_cstr = noofpostconting_cstr+1;
                        count_post = count_post + 1;
                    end
                end
            end
        end

        Pivot = Pivot+noofpostconting_cstr;
        X_val = Xd+noofpostconting_cstr*ones(length(Xd),1);
```

```
    Xd = vertcat(Xd,X_val);
    Yd = repmat(Yd,2,1);
    Vd = repmat(Vd,2,1);
    Pivot = Pivot+noofpostconting_cstr;

%% Constraints for cutset power trasnfer:
    [row_K, col_K] = size(K_ft_unique);
    count_Se = 1;
    Xe = [];
    Ye = [];
    Ve = [];
    Rhs_cutset = [];
    Sign_cutset = [];

    for ncutset = 1:row_K
        for ngen = 1:noofgen
            GenBusNum = Generator(ngen,1);
            PTDF_cutset = 0;
            for nbranch = 1:col_K
                if K_ft_unique(ncutset,nbranch)~=0
                    BranchNum = K_ft_unique(ncutset,nbranch);
                    % Check if the direction of the branch is same the
direction
                    % of the cut-set.
                    F_Branch = Branch(BranchNum,1);
                    T_Branch = Branch(BranchNum,2);
                    Sign = 0;
                    if IsPresent(Cutset_FT(:,1,ncutset),F_Branch)==1 &&
IsPresent(Cutset_FT(:,2,ncutset),T_Branch)==1
                        Sign = 1;
                    elseif IsPres-
ent(Cutset_FT(:,2,ncutset),F_Branch)==1 && IsPres-
ent(Cutset_FT(:,1,ncutset),T_Branch)==1
                        Sign = -1;
                    else
                        Sign = 0;
                    end
                    if (GenBusNum < length(Bus(:,1)))
                        PTDF_val = Sign*PTDF(BranchNum,GenBusNum);
                    else
                        PTDF_val = 0;
                    end
                    PTDF_cutset = PTDF_cutset+PTDF_val;
                end
            end
           if (PTDF_cutset~=0)
               Xe(count_Se,1) = Pivot+ncutset;
               Ye(count_Se,1) = ngen;
               Ve(count_Se,1) = PTDF_cutset;
               count_Se = count_Se + 1;
           end
        end
        for nload = 1:noofload
            LoadBusNum = Load(nload,1);
            PTDF_cutset = 0;
```

```
            for nbranch = 1:col_K
                if K_ft_unique(ncutset,nbranch)~=0
                    BranchNum = K_ft_unique(ncutset,nbranch);
                    F_Branch = Branch(BranchNum,1);
                    T_Branch = Branch(BranchNum,2);
                    Sign = 0;
                    if (IsPresent(Cutset_FT(:,1,ncutset),F_Branch)==1
&& IsPresent(Cutset_FT(:,2,ncutset),T_Branch)==1)
                        Sign = 1;
                    elseif (IsPres-
ent(Cutset_FT(:,2,ncutset),F_Branch)==1 && IsPres-
ent(Cutset_FT(:,1,ncutset),T_Branch)==1)
                        Sign = -1;
                    else
                        Sign = 0;
                    end
                    if (LoadBusNum < length(Bus(:,1)))
                        PTDF_val = Sign*PTDF(BranchNum,LoadBusNum);
                    else
                        PTDF_val = 0;
                    end
                    PTDF_cutset = PTDF_cutset+(-1)*PTDF_val;
                end
            end
            if (PTDF_cutset~=0)
                Xe(count_Se,1) = Pivot+ncutset;
                Ye(count_Se,1) = noofgen+nload;
                Ve(count_Se,1) = PTDF_cutset;
                count_Se = count_Se + 1;
            end
        end

        Tot_rate = 0;
        Tot_flow = 0;
        for i = 1:length(K_ft_unique(ncutset,:))
            if K_ft_unique(ncutset,i)==0
                break;
            end
            if (i>1)
                Tot_rate = Tot_rate+Branch(K_ft_unique(ncutset,i),7);
            end
            A_Branch = Branch(K_ft_unique(ncutset,i),1);
            B_Branch = Branch(K_ft_unique(ncutset,i),2);
            if (IsPresent(Cutset_FT(:,1,ncutset),A_Branch)==1) && (Is-
Present(Cutset_FT(:,2,ncutset),B_Branch)==1)
                Tot_flow = Tot_flow + Branch(K_ft_unique(ncutset,i),6);
            else
                Tot_flow = Tot_flow + (-
1)*Branch(K_ft_unique(ncutset,i),6);
            end
        end
        Rhs_cutset(ncutset,1) = Tot_rate-Tot_flow;
        Sign_cutset(ncutset,1) = '<';
    end
    Pivot = Pivot + row_K;
```

```
    X = vertcat(Xa,Xb,Xc,Xd,Xe);
    Y = vertcat(Ya,Yb,Yc,Yd,Ye);
    V = vertcat(Va,Vb,Vc,Vd,Ve);
    T = horzcat(X,Y,V);

    Constraint_SP = sparse(X,Y,V);

    %% Combine all the Constraint Matrices Together:
    RHS = vertcat(Rhs_con-
serve,Rhs_pinj_max,Rhs_pinj_min,Rhs_MaxFlow,Rhs_Min-
Flow,Rhs_MaxFlow_post,Rhs_MinFlow_post,Rhs_cutset);
    SIGN = vertcat(Sign_con-
serve,Sign_pinj_max,Sign_pinj_min,Sign_Maxflow,Sign_Min-
flow,Sign_Maxflow_post,Sign_Minflow_post,Sign_cutset);

    %% Use the quadratic cost coefficients:
    f_quad_gen = zeros(noofgen+noofload);
    for i = 1:noofgen
        c_quad = c(i,1);
        f_quad_gen(i,i) = c_quad; % Additional soft constraint on
delta_Pgi
    end

    %% Set the model parameters:
    model.obj = f;
    model.Q = sparse(f_quad_gen); % Include quadratic cost coefficients
    model.A = Constraint_SP;
    model.sense = SIGN;
    model.rhs = RHS;
    model.lb = Rhs_pinj_min;

    clear params;
    params.outputflag = 0;
    result = gurobi(model, params);

    if ((strcmp(result.status,'OPTIMAL')==1) || (strcmp(result.sta-
tus,'SUBOPTIMAL')==1))
        Soln_Flag = 1;
        xf = result.x;
        %% Compute all measurement values after solving the optmiza-
tion:
        % New branch flows:
        flow_old = zeros(length(Branch(:,1)),4);
        flow_new = zeros(length(Branch(:,1)),4);
        flow_old(:,1) = Branch(:,1);
        flow_old(:,2) = Branch(:,2);
        flow_old(:,3) = Branch(:,6);
        flow_old(:,4) = Branch(:,7);
        delta_flow = Constraint_flow*xf;
        flow_new(:,1) = Branch(:,1);
        flow_new(:,2) = Branch(:,2);
        flow_new(:,3) = flow_old(:,3)+delta_flow;
        flow_new(:,4) = Branch(:,7);
```

```
        % New dispatch:
        gen_old = zeros(length(Generator(:,1)),2);
        gen_new = zeros(length(Generator(:,1)),2);
        gen_old(:,1) = Generator(:,1);
        gen_old(:,2) = Generator(:,2);
        load_old(:,1) = Load(:,1);
        load_old(:,2) = Load(:,2);
        delta_inj = Constraint_pinj*xf;
        delta_pgen = delta_inj([1:noofgen],1);
        delta_pload = delta_inj([noofgen+1:noofgen+noofload],1);

        Generator_New(:,1) = Generator(:,1);
        Generator_New(:,2) = gen_old(:,2)+delta_pgen;
        Load_New(:,1) = Load(:,1);
        Load_New(:,2) = load_old(:,2)+delta_pload;

        %% Finding the actual cost using quadratic and linear cost co-
efficients:
        cost_linear = transpose(f);
        cost_quad = horzcat(transpose(c),zeros(1,noofload));
        tot_change_cost = cost_linear*xf + cost_quad*(xf.^2);

        %% Find the positions where non-zero changes have occurred in
Pgen:
        [ indpos, ~ ] = find(delta_pgen>0.001);
        [ indneg, ~ ] = find(delta_pgen<-0.001);
        GeneratorPositiveChange(:,1) = Generator(indpos,1);
        GeneratorPositiveChange(:,2) = delta_pgen(indpos);

        GeneratorNegativeChange(:,1) = Generator(indneg,1);
        GeneratorNegativeChange(:,2) = delta_pgen(indneg);

        %% Find the positions where non-zero changes have occurred in
Pload:
        [ indpos, ~ ] = find(delta_pload>0.001);
        [ indneg, ~ ] = find(delta_pload<-0.001);
        LoadPositiveChange(:,1) = Load(indpos,1);
        LoadPositiveChange(:,2) = delta_pload(indpos);

        LoadNegativeChange(:,1) = Load(indneg,1);
        LoadNegativeChange(:,2) = delta_pload(indneg);


        %% Get the data for the next stage:
        % Get the new flows for the branch:
        Branch(:,6) = flow_new(:,3);
        % Get the new generation values:
        Generator(:,2) = Generator_New(:,2);
        % Get the new load values:
        Load(:,2) = Load_New(:,2);


    else
        Soln_Flag = 0;
```

```
        GeneratorNegativeChange = [];
        GeneratorPositiveChange = [];
        LoadNegativeChange = [];
        LoadPositiveChange = [];
        tot_change_cost = 0;
    end

    %% Print the change in dispatches on the screen:
   if (Soln_Flag==1)
        fprintf('------------------------------------------ \n');
        fprintf('Total amount of load shed = %f \n',round(sum(LoadNega-
tiveChange(:,2))));
        fprintf('Total increase in dispatch  = %f \n', round(sum(Gener-
atorPositiveChange(:,2))));
        fprintf('Total decrease in dispatch  = %f \n', round(sum(Gener-
atorNegativeChange(:,2))));
        fprintf('Total change in cost of generation = $ %f
\n',round(tot_change_cost));
        fprintf('------------------------------------------ \n');
   else
        fprintf('------------------------------------------ \n');
        fprintf('No feasible solution obtained! \n');
        fprintf('------------------------------------------ \n');
   end
   time = toc;

end
```

```
function [ flag, pos ] = IsPresent( Arr, Val )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                             %
% Program Description: Checks if a given number is contained                  %
% in a specific array                                                         %
%                                                                             %
% Author: Reetam Sen Biswas                                                   %
% Arizona State University                                                    %
%                                                                             %
% Last Modified: 03/20/2020                                                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

flag = 0;
pos = 0;
for i = 1:length(Arr)
    if Val==Arr(i)
        flag = 1;
        pos = i;
        break;
    end
end


end
```

```
function [ Shortlist, time ] = ModifiedShortlistAssets( Branch-
FlowChange, EdgeList, Branch )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                             %
% Program Description: This program shortlists the transmission              %
% assets that must be re-evaluated by the feasibility test (FT)               %
% algorithm following a generation redispatch in the system.                  %
% This logic for this program is based on the M-SA algorithm.                 %
%                                                                             %
% Author: Reetam Sen Biswas                                                   %
% Arizona State University                                                    %
%                                                                             %
% Last Modified: 03/20/2020                                                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

tic;
len = length(BranchFlowChange);
[row_lim,col_lim] = size(EdgeList);
countF = 1;
Shortlist = [];

for r = 1:row_lim
    Common = 0;
    if (EdgeList(r,1)~=0)
        for i = 1:2:len
            F = BranchFlowChange(1,i);
            T = BranchFlowChange(1,i+1);

            col_lim = length(find(EdgeList(r,:)~=0)); % This line is
newly added
            for c = 1:2:col_lim-1
                if (F==EdgeList(r,c) && T==EdgeList(r,c+1)) ||
(F==EdgeList(r,c+1) && T==EdgeList(r,c))
                    Common = 1;
                    break;
                end
            end
            if (F==Branch(r,1) && T==Branch(r,2)) || (F==Branch(r,2) &&
T==Branch(r,1))
                Common = 1;
            end
            if Common==1
                break;
            end
        end
        if (Common==1)
            Shortlist(countF,1) = r;
            Shortlist(countF,2) = Branch(r,1);
            Shortlist(countF,3) = Branch(r,2);
            countF = countF + 1;
        end
    end
end
time = toc;
end
```

```
function [ Flow, Capacity, BranchFlowChange, time ] = Modi-
fiedUpdateScheme( Flow, Capacity, GeneratorPositiveChange, Genera-
torNegativeChange, Branch )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                             %
% Program Description: This program creates an updated "flow"                %
% and "latent capacity graph" after change in generation                      %
% in the system. The logic for this program is based on the                   %
% M-UPS algorithm.                                                            %
%                                                                             %
% Author: Reetam Sen Biswas                                                   %
% Arizona State University                                                    %
%                                                                             %
% Last Modified: 03/20/2020                                                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 tic;
% Initialization:
    DontSelect = [];
    countD = 1;
    problem = 0;
    NoOfBFS = 0;
    EdgesFlowChange = [];
    count = 1;
    countBT = 1;
    BranchFlowChange = [];
    countChange = 1;
    BranchTouch = [];
    FlowOrg = Flow;
    CapacityOrg = Capacity;
    GenIncData = GeneratorPositiveChange;
    GenDecData = GeneratorNegativeChange;
    GenPosInc = GenIncData(:,1);GenInc = GenIncData(:,2);
    GenPosDec = GenDecData(:,1);GenDec = abs(GenDecData(:,2));

% Select Source-Sink pairs from generator increase and decrease pairs
to update the flow and capacity graphs:
    while (1<2)
      FF = CheckZeros(GenDec);
      GG = CheckZeros(GenInc);
      if (Is_i_Present(0,FF)==1) || (Is_i_Present(0,GG)==1)
      else
        break;
      end
      if (sum(GenDec)<0.01 && sum(GenInc)<0.01)
        break;
      end

      for i = 1:length(GenDec)
          if GenDec(i)~=0
              Sink = GenPosDec(i);
              break;
          end
      end

     %% Selection of the source:
```

```
    % Select a "source" depending upon the position of the "sink"
       for j = 1:length(GenInc)
           if GenInc(j)~=0 && problem==0
               Source = GenPosInc(j);
               break;
           else
               if GenInc(j)~=0 && IsPresent(DontSelect,j)==0
                   Source = GenPosInc(j);
                   break;
               end
           end
       end

% Finding the shortest path from the Source to the Sink and finding out
the maximum capacity of the path.
       while (1<2)
           [S,path]=graphshortestpath(Capac-
ity,Source,Sink,'Method','BFS','Directed','true');NoOfBFS = NoOfBFS +
1;
           if S<Inf
               for ii = 1:S
                   F = path(ii);T=path(ii+1);
                   len = length(BranchFlowChange);
                   if len>0
                       Present = 0;
                       for kk = 1:2:len
                           Fbr = BranchFlowChange(1,kk);
                           Tbr = BranchFlowChange(1,kk+1);
                           if (F==Fbr && T==Tbr) || (F==Tbr && T==Fbr)
                               Present = 1;
                               break;
                           end
                       end
                       if Present==0
                           BranchFlowChange(1,countChange) = F;
                           BranchFlowChange(1,countChange+1) = T;
                           countChange = countChange+2;
                       end
                   else
                       BranchFlowChange(1,countChange) = F;
                       BranchFlowChange(1,countChange+1) = T;
                       countChange = countChange+2;
                   end
               end
           end
           if ((S==Inf) && (IfCloseToZero(GenDec(i))==0) &&
(IfCloseToZero(GenInc(j))==0))
               problem = 1;
               DontSelect(countD) = j; countD = countD+1;
           else
               problem = 0;
               DontSelect = [];
               countD = countD+1;
           end
           if ((S==Inf) || (GenDec(i)==0) || (GenInc(j)==0))
               break;
```

```
            end
            MaxCap = 999999;
            for k=1:S
                From = path(k);To = path(k+1);
                if MaxCap>Capacity(From,To)
                    MaxCap = Capacity(From,To);
                end
            end
% Determine the flow injection along a given path
            if ((GenDec(i)<=MaxCap) && (GenDec(i)<=GenInc(j)))
                FlowInj = GenDec(i);
            elseif ((GenInc(j)<=MaxCap) && (GenInc(j)<=GenDec(i)))
                FlowInj = GenInc(j);
            elseif ((MaxCap<=GenDec(i)) && (MaxCap<=GenInc(j)))
                FlowInj = MaxCap;
            end
% Update the load and generation values
            GenDec(i) = GenDec(i)-FlowInj;
            GenInc(j) = GenInc(j)-FlowInj;
% Update the "flow" and "latent capacity" graphs for
% power injection along the given path
            for k=1:S
                From = path(k);To = path(k+1);
                Flow(From,To) = Flow(From,To) + FlowInj;
                Flow(To,From) = Flow(To,From) - FlowInj;
                Capacity(From,To) = Capacity(From, To) - FlowInj;
                Capacity(To,From) = Capacity(To, From) + FlowInj;
            end
        end
    end
time = toc;
end
```

```
function [ Flow, Capacity, A, time ] = NetworkFlowAlgorithm( Bus,
Branch, Gen, BusLoad )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                             %
% Program Description: The network flow algorithm (NFA) creates the          %
% "flow" and "latent capacity graphs" based upon the                          %
% conservation of energy.                                                     %
%                                                                             %
% Author: Reetam Sen Biswas                                                   %
% Arizona State University                                                    %
%                                                                             %
% Last Modified: 03/20/2020                                                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

tic;
%% Initialize arrays and variables
NoOfBus = length(Bus);
NoOfBranch = length(Branch);
BranchSt = ones(NoOfBranch,1);

GenPos = Gen(:,1);
Generation = Gen(:,2);
LoadPos = BusLoad(:,1);
Load = BusLoad(:,2);

NoOfBFS = 0;
problem = 0;

%% Initialize the "flow" and "latent capacity" graphs
Capacity = sparse(NoOfBus,NoOfBus); % Latent capacity graph
Flow = sparse(NoOfBus,NoOfBus); % Flow graph
A = sparse(NoOfBus,NoOfBus); % Incidence matrix
for k=1:NoOfBranch
     if BranchSt(k)==1
        Capacity(Branch(k,1),Branch(k,2)) = Capac-
ity(Branch(k,1),Branch(k,2)) + Branch(k,3);
        Capacity(Branch(k,2),Branch(k,1)) = Capac-
ity(Branch(k,2),Branch(k,1)) + Branch(k,3);
        A(Branch(k,1),Branch(k,2)) = 1; A(Branch(k,1),Branch(k,1)) = 1;
        A(Branch(k,2),Branch(k,1)) = 1; A(Branch(k,2),Branch(k,2)) = 1;
     end
end
DontSelect = [];
countD = 1;

%% Create the "flow" and "latent capacity" graphs iteratively
while (1<2)
      FF = CheckZeros(Load);
      GG = CheckZeros(Generation);
      if (Is_i_Present(0,FF)==1) || (Is_i_Present(0,GG)==1)
      else
        break;
      end

      for i = 1:length(Load)
```

```
            if Load(i)~=0
                Sink = LoadPos(i);
                break;
            end
        end

      % Selection of the source:
         for j = 1:length(Generation)
             if Generation(j)~=0 && problem==0
                 Source = GenPos(j);
                 break;
             else
                 if Generation(j)~=0 && IsPresent(DontSelect,j)==0
                     Source = GenPos(j);
                     break;
                 end
             end
         end

         % Finding the maximum power that can be injected along
         % the shortest path from the source to the sink
         while (1<2)
             [S,path]=graphshortestpath(Capac-
ity,Source,Sink,'Method','BFS','Directed','true');NoOfBFS = NoOfBFS +
1;
             if S==Inf && IfCloseToZero(Load(i))==0 &&
IfCloseToZero(Generation(j))==0
                 problem = 1;
                 DontSelect(countD) = j; countD = countD+1;
             else
                 problem = 0;
                 DontSelect = [];
                 countD = countD+1;
             end
             if S==Inf || Load(i)==0 || Generation(j)==0
                 break;
             end
             MaxCap = 999999;
             for k=1:S
                 From = path(k);To = path(k+1);
                 if MaxCap>Capacity(From,To)
                     MaxCap = Capacity(From,To);
                 end
             end

         % Determining the flow that will be injected along the path
             if Load(i)<=MaxCap && Load(i)<=Generation(j)
                 FlowInj = Load(i);
             elseif Generation(j)<=MaxCap && Generation(j)<=Load(i)
                 FlowInj = Generation(j);
             elseif MaxCap<=Load(i) && MaxCap<=Generation(j)
                 FlowInj = MaxCap;
             end

         % Updating the source and sink values:
```

```
            Load(i) = Load(i)-FlowInj;
            Generation(j) = Generation(j)-FlowInj;

        % Updating the "flow" and "latent capacity" graph based upon
the power
        % transferred along different paths
            for k=1:S
                From = path(k);To = path(k+1);
                Flow(From,To) = Flow(From,To) + FlowInj;
                Flow(To,From) = Flow(To,From) - FlowInj;
                Capacity(From,To) = Capacity(From, To) - FlowInj;
                Capacity(To,From) = Capacity(To, From) + FlowInj;
            end
       end
end

time = toc;
end
```

```
function [ Flow, Capacity, A CL_Sp, EdgeList, PathStack, EdgeSatStack,
CutsetStack, time ] = OutageAnalysis( Branch, Flow, Capacity, LineOut-
Number, EdgeList, A )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                       %
% Program Description: This program performs feasibility test (FT)     %
% following a branch outage in the system. Therefore, it involves       %
% the following:                                                        %
% (a) The Update Scheme (UPS) for updating the weighted graphs          %
%     after the outage                                                  %
% (b) The Shortlisting Assets (SA) algorithm to determine the           %
%     assets which should be evaluated by FT                            %
% (c) The feasibility test (FT) on the shortlisted set of assets        %
%                                                                       %
% Author: Reetam Sen Biswas                                             %
% Arizona State University                                              %
%                                                                       %
% Last Modified: 03/20/2020                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tic;
    [ LoseFlag, PathAr, CurrentFlow, FlowCap, FlowInjAr ] = CheckI-
fLose_Cutset( Branch, LineOutNumber, Flow, Capacity, A );
%% The Update Scheme (UPS):
    if LoseFlag==1
        [  Flow, Capacity ] = UpdateScheme( Branch, LineOutNumber,
Flow, Capacity );
        A(Branch(LineOutNumber,1),Branch(LineOutNumber,2)) = 0;
        A(Branch(LineOutNumber,2),Branch(LineOutNumber,1)) = 0;
    else
        fprintf('\n Warning! Outage of the branch saturates a cut-set.
\n');
    end

%% The Shortlisting Assets (SA):
    Shortlist = ShortlistAssets( Branch, EdgeList, LineOutNumber );
 % The branches with zero latent capacities are indentified, and
% the latent capacities are increased by a small margin to ensure that
all
% cut-sets are identified properly
    for Line = 1:length(Branch(:,1))
        if Capacity(Branch(Line,1),Branch(Line,2))==0 &&
Branch(Line,4)==1
            Capacity(Branch(Line,1),Branch(Line,2)) = 0.0001;
        elseif Capacity(Branch(Line,2),Branch(Line,1))==0 &&
Branch(Line,4)==1
            Capacity(Branch(Line,2),Branch(Line,1)) = 0.0001;
        end
    end
 %% Feasibility Test (FT) on shortlisted assets:
    CL_Sp = [];
    PathStack = [];
    EdgeSatStack = [];
    CutsetStack = [];

    count = 1;
```

```
    [rowF, colF] = size(Shortlist);
    for i=1:rowF
        Line = Shortlist(i,1);
        FlagPresBefore = 0;
        if FlagPresBefore==0
            [ LoseFlag, PathAr, CurrentFlow, FlowCap, FlowInjAr,
flag_Radial, EdgeSat, Cutset ] = CheckIfLose_Cutset( Branch, Line,
Flow, Capacity, A );
            [ row, col ] = size(PathAr);
            EdgeCount = 1;
            EnterLoop = 0;
            EdgeList(Line,:) = zeros(1,length(EdgeList(Line,:)));
            for R = 1:row
                for C = 1:col-1
                    if PathAr(R,C+1)>0
                        PresentFlag = 0;
                        if EnterLoop==1
                            Col_list = length(EdgeList(Line,:));
                            for k = 1:Col_list-1
                                if EdgeList(Line,k)==PathAr(R,C) &&
EdgeList(Line,k+1)==PathAr(R,C+1)
                                    PresentFlag=1;
                                end
                            end
                        end

                        if PresentFlag==0
                            EdgeList(Line,EdgeCount) = PathAr(R,C);
                            EdgeList(Line,EdgeCount+1) = PathAr(R,C+1);
                            EdgeCount = EdgeCount+2;
                        end
                        EnterLoop = 1;
                    end
                end
            end
            [row, col] = size(PathAr);
            [row_e, col_e] = size(EdgeSat);
            [row_K,col_K] = size(Cutset);
            if (LoseFlag==0) && (flag_Radial==0)
                PathInterest([1:row],[1:col],count) = PathAr;
                CL_Sp(count,1) = Line;
                CL_Sp(count,2) = Branch(Line,1);
                CL_Sp(count,3) = Branch(Line,2);
                CL_Sp(count,4) = FlowCap-CurrentFlow;
                CL_Sp(count,5) = flag_Radial;
                PathStack(1:row,1:col,count) = PathAr;
                EdgeSatStack(1:row_e,1:col_e,count) = EdgeSat;
                CutsetStack([1:row_K],[1:col_K],count) = Cutset;
                count = count + 1;
            end
        end
    end
time = toc;
end
```

```
function [ GeneratorNegativeChange, GeneratorPositiveChange, LoadNega-
tiveChange, LoadPositiveChange, Branch, Load, Generator, Soln_Flag,
Flow_dc, Rate_dc, tot_change_cost, time ] = RelaxedCorrectiveAction( K,
Tm, Cutset_FT, PTDF, Bus, Branch, Generator, Load)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                       %
% Program Description: This program solves the optimization             %
% problem for the relaxed corrective action (rCA) used                  %
% in the second component                                               %
%                                                                       %
% Author: Reetam Sen Biswas                                             %
% Arizona State University                                              %
%                                                                       %
% Last Modified: 03/20/2020                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


    tic;
    % Find the length of different arrays related to the cut-set viola-
tions
    NumOfCritCutset = length(K(:,1));
    MaxNumBranchCritCutset = length(K(1,:));
    [numrow_K, numcol_K, numsheet] = size(Cutset_FT);

    %% Initialize flow across different cutsets:
    flow_cutset = [];
    for snum = 1:numsheet
        flow_total = 0;
        for rnum = 1:numrow_K
            if Cutset_FT(rnum,1,snum)~=0
                FromBus = Cutset_FT(rnum,1,snum);
                ToBus = Cutset_FT(rnum,2,snum);
                for i = 1:length(Branch(:,1))
                    if (FromBus==Branch(i,1) && ToBus==Branch(i,2))
                        flow_val = Branch(i,6);
                        flow_total = flow_total+flow_val;
                    elseif (FromBus==Branch(i,2) && ToBus==Branch(i,1))
                        flow_val = (-1)*Branch(i,6);
                        flow_total = flow_total+flow_val;
                    end
                end
            end
            flow_cutset(snum,1) = flow_total;
        end
    end
 %% Set-up the objective function:
    b = Generator(:,5); % The linear cost coefficient
    c = Generator(:,6); % The quadratic cost coefficient
    Pg_old_ar = Generator(:,2); % Old power generation
    f_gen_lin = (2*(Pg_old_ar.*c) + b);
    f_load = Load(:,3);
    f = vertcat(f_gen_lin,f_load);

    noofline = length(Branch(:,1));
```

```
    noofgen = length(Generator(:,1));
    noofload = length(Load(:,1));

%% Constraints for the conservation of energy:
    Pivot = 1;
    count_Sa = 1;
    for i = 1:noofgen
       Xa(count_Sa,1) = Pivot;Ya(count_Sa,1) = i;Va(count_Sa,1) = 1;
       count_Sa = count_Sa + 1;
    end
    for i = noofgen+1:(noofgen+noofload)
       Xa(count_Sa,1) = Pivot;Ya(count_Sa,1) = i;Va(count_Sa,1) = -1;
       count_Sa = count_Sa + 1;
    end
    Rhs_conserve = [0];
    Sign_conserve = [ '=' ];

     %% Constraints for the injection limits:
    Constraint_pinj = eye(noofgen+noofload,noofgen+noofload);
    count_Sb = 1;
    for i = 1:(noofgen+noofload)
         Xb(count_Sb,1) = Pivot+i;Yb(count_Sb,1) = i;Vb(count_Sb,1) =
1;
        count_Sb = count_Sb+1;
    end
    Pivot = Pivot+noofgen+noofload;
    X_val = Pivot+1:Pivot+noofgen+noofload; X_val = X_val';
    Xb = vertcat(Xb,X_val);
    Yb = repmat(Yb,2,1);
    Vb = repmat(Vb,2,1);
    Pivot = Pivot+noofgen+noofload;
    for ngen = 1:noofgen
        GenBusNum = Generator(ngen,1);
        Pgen_old = Generator(ngen,2);
        Pgen_max = Generator(ngen,3);
        Pgen_min = Generator(ngen,4);
        Rhs_pinj_max(ngen,1) = Pgen_max-Pgen_old;
        Rhs_pinj_min(ngen,1) = Pgen_min-Pgen_old;
        Sign_pinj_max(ngen,1) = '<';
        Sign_pinj_min(ngen,1) = '>';
    end

    %% LHS and RHS for the injection limits:
    for nload = 1:noofload
        LoadBusNum = Load(nload,1);
        Pload_old = Load(nload,2);
        Rhs_pinj_max(noofgen+nload,1) = 0;
        Rhs_pinj_min(noofgen+nload,1) = -Pload_old;
        Sign_pinj_max(noofgen+nload,1) = '<';
        Sign_pinj_min(noofgen+nload,1) = '>';
    end

    %% Constraints for pre-contingency power flow in each branch:
    Constraint_flow = zeros(noofline,noofgen+noofload);
    count_Sc = 1;
```

```
    noofflow_cstr = 0;
    for nline = 1:noofline
        flag_flow = 0;
        if (Branch(nline,8)==1)
            for ngen = 1:noofgen
                GenBusNum = Generator(ngen,1);
                if (GenBusNum==length(Bus))
                    PTDF_val = 0;
                else
                    PTDF_val = PTDF(nline,GenBusNum);
                end
                if abs(PTDF_val)>10^-5
                    Constraint_flow(nline,ngen) = PTDF_val;
                    Xc(count_Sc,1) = Pivot+noofflow_cstr+1;
Yc(count_Sc,1) = ngen; Vc(count_Sc,1) = PTDF_val;
                    count_Sc = count_Sc+1;
                    flag_flow = 1;
                end
            end
            for nload = 1:noofload
                LoadBusNum = Load(nload,1);
                if (LoadBusNum==length(Bus))
                    PTDF_val = 0;
                else
                    PTDF_val = PTDF(nline,LoadBusNum);
                end
%                 if (PTDF_val~=0)
                if abs(PTDF_val)>10^-5
                    Constraint_flow(nline,noofgen+nload) = (-
1)*PTDF_val;
                    Xc(count_Sc,1) = Pivot+noofflow_cstr+1;
Yc(count_Sc,1) = noofgen+nload; Vc(count_Sc,1) = (-1)*PTDF_val;
                    count_Sc = count_Sc+1;
                    flag_flow = 1;
                end
            end
            if (flag_flow==1)
                flow_old = Branch(nline,6);
                flow_max = Branch(nline,7);
                flow_min = (-1)*Branch(nline,7);
                Rhs_MaxFlow(noofflow_cstr+1,1) = flow_max-flow_old;
                Rhs_MinFlow(noofflow_cstr+1,1) = flow_min-flow_old;
                Sign_Maxflow(noofflow_cstr+1,1) = '<';
                Sign_Minflow(noofflow_cstr+1,1) = '>';
                noofflow_cstr = noofflow_cstr + 1;
            end
        end
    end
    Pivot = Pivot + noofflow_cstr;
    X_val = Xc+noofflow_cstr*ones(length(Xc),1);
    Xc = vertcat(Xc,X_val);
    Yc = repmat(Yc,2,1);
    Vc = repmat(Vc,2,1);
    Pivot = Pivot + noofflow_cstr;

    %% Constraints for cut-set power transfer limit:
```

```
      %% Constraints for the power transfer across the cut-set:
    [row_K, col_K] = size(K);
    count_Sd = 1;
    for ncutset = 1:row_K
        for ngen = 1:noofgen
            GenBusNum = Generator(ngen,1);
            PTDF_cutset = 0;
            for nbranch = 1:col_K
                if K(ncutset,nbranch)~=0
                    BranchNum = K(ncutset,nbranch);
                    % Check if the direction of the branch is same the
direction
                    % of the cut-set.
                    F_Branch = Branch(BranchNum,1);
                    T_Branch = Branch(BranchNum,2);
                    Sign = 0;
                    if IsPresent(Cutset_FT(:,1,ncutset),F_Branch)==1 &&
IsPresent(Cutset_FT(:,2,ncutset),T_Branch)==1
                        Sign = 1;
                    elseif IsPres-
ent(Cutset_FT(:,2,ncutset),F_Branch)==1 && IsPres-
ent(Cutset_FT(:,1,ncutset),T_Branch)==1
                        Sign = -1;
                    else
                        Sign = 0;
                    end
                    if (GenBusNum < length(Bus(:,1)))
                        PTDF_val = Sign*PTDF(BranchNum,GenBusNum);
                    else
                        PTDF_val = 0;
                    end
                    PTDF_cutset = PTDF_cutset+PTDF_val;
                end
            end
           if abs(PTDF_cutset)>10^-5
               Xd(count_Sd,1) = Pivot+ncutset;
               Yd(count_Sd,1) = ngen;
               Vd(count_Sd,1) = PTDF_cutset;
               count_Sd = count_Sd + 1;
           end
        end
        for nload = 1:noofload
            LoadBusNum = Load(nload,1);
            PTDF_cutset = 0;
            for nbranch = 1:col_K
                if K(ncutset,nbranch)~=0
                    BranchNum = K(ncutset,nbranch);
                    % Check if the direction of the branch is same the
direction
                    % of the cut-set.
                    F_Branch = Branch(BranchNum,1);
                    T_Branch = Branch(BranchNum,2);
                    Sign = 0;
                    if IsPresent(Cutset_FT(:,1,ncutset),F_Branch)==1 &&
IsPresent(Cutset_FT(:,2,ncutset),T_Branch)==1
                        Sign = 1;
```

```
                    elseif IsPres-
ent(Cutset_FT(:,2,ncutset),F_Branch)==1 && IsPres-
ent(Cutset_FT(:,1,ncutset),T_Branch)==1
                        Sign = -1;
                    else
                        Sign = 0;
                    end
                    if (LoadBusNum < length(Bus(:,1)))
                        PTDF_val = Sign*PTDF(BranchNum,LoadBusNum);
                    else
                        PTDF_val = 0;
                    end
                    PTDF_cutset = PTDF_cutset+(-1)*PTDF_val;
                end
            end
            if abs(PTDF_cutset)>10^-5
                Xd(count_Sd,1) = Pivot+ncutset;
                Yd(count_Sd,1) = noofgen+nload;
                Vd(count_Sd,1) = PTDF_cutset;
                count_Sd = count_Sd + 1;
            end
        end

        Tot_rate = 0;
        Tot_flow = 0;
        for i = 1:length(K(ncutset,:))
            if K(ncutset,i)==0
                break;
            end
            if (i>1)
                Tot_rate = Tot_rate+Branch(K(ncutset,i),7);
            end
            A_Branch = Branch(K(ncutset,i),1);
            B_Branch = Branch(K(ncutset,i),2);
            if (IsPresent(Cutset_FT(:,1,ncutset),A_Branch)==1) && (Is-
Present(Cutset_FT(:,2,ncutset),B_Branch)==1)
                Tot_flow = Tot_flow + Branch(K(ncutset,i),6);
            else
                Tot_flow = Tot_flow + (-1)*Branch(K(ncutset,i),6);
            end
        end
        Rhs_cutset(ncutset,1) = Tot_rate-Tot_flow;
        Sign_cutset(ncutset,1) = '<';
    end
    Pivot = Pivot + row_K;

    %% Concatenate all constraints:
    X = vertcat(Xa,Xb,Xc,Xd);
    Y = vertcat(Ya,Yb,Yc,Yd);
    V = vertcat(Va,Vb,Vc,Vd);
    Constraint_SP = sparse(X,Y,V);
    RHS = vertcat(Rhs_con-
serve,Rhs_pinj_max,Rhs_pinj_min,Rhs_MaxFlow,Rhs_MinFlow,Rhs_cutset);
```

```
    SIGN = vertcat(Sign_con-
serve,Sign_pinj_max,Sign_pinj_min,Sign_Maxflow,Sign_Min-
flow,Sign_cutset);
    %% Use the quadratic cost coefficients:
    f_quad_gen = zeros(noofgen+noofload);
    for i = 1:noofgen
        c_quad = c(i,1);
        f_quad_gen(i,i) = c_quad;
    end

    %% Set the model parameters:
    model.obj = f;
    model.Q = sparse(f_quad_gen);
    model.A = sparse(Constraint_SP);
    model.sense = SIGN;
    model.rhs = RHS;
    model.lb = Rhs_pinj_min;
    clear params;
    params.outputflag = 0;
    result = gurobi(model, params);

    if length(result.status)==7
        Soln_Flag = 1;
        xf = result.x;
        %% Compute all measurement values after solving the optmiza-
tion:
        % New branch flows:
        flow_old = zeros(length(Branch(:,1)),4);
        flow_new = zeros(length(Branch(:,1)),4);
        flow_old(:,1) = Branch(:,1);
        flow_old(:,2) = Branch(:,2);
        flow_old(:,3) = Branch(:,6);
        flow_old(:,4) = Branch(:,7);
        delta_flow = Constraint_flow*xf;
        flow_new(:,1) = Branch(:,1);
        flow_new(:,2) = Branch(:,2);
        flow_new(:,3) = flow_old(:,3)+delta_flow;
        flow_new(:,4) = Branch(:,7);
        % New dispatch:
        gen_old = zeros(length(Generator(:,1)),2);
        gen_new = zeros(length(Generator(:,1)),2);
        gen_old(:,1) = Generator(:,1);
        gen_old(:,2) = Generator(:,2);
        load_old(:,1) = Load(:,1);
        load_old(:,2) = Load(:,2);
        delta_inj = Constraint_pinj*xf;
        delta_pgen = delta_inj([1:noofgen],1);
        delta_pload = delta_inj([noofgen+1:noofgen+noofload],1);
        Generator_New(:,1) = Generator(:,1);
        Generator_New(:,2) = gen_old(:,2)+delta_pgen;
        Load_New(:,1) = Load(:,1);
        Load_New(:,2) = load_old(:,2)+delta_pload;

        %% Finding the actual cost using quadratic and linear cost co-
efficients:
```

```
        cost_linear = transpose(f);
        cost_quad = horzcat(transpose(c),zeros(1,noofload));
        tot_change_cost = cost_linear*xf + cost_quad*(xf.^2);

        %% Find the positions where non-zero changes have occurred in
Pgen:
        [ indpos, ~ ] = find(delta_pgen>0.001);
        [ indneg, ~ ] = find(delta_pgen<-0.001);
        GeneratorPositiveChange(:,1) = Generator(indpos,1);
        GeneratorPositiveChange(:,2) = delta_pgen(indpos);

        GeneratorNegativeChange(:,1) = Generator(indneg,1);
        GeneratorNegativeChange(:,2) = delta_pgen(indneg);
  %% Find the positions where non-zero changes have occurred in Pload:
        [ indpos, ~ ] = find(delta_pload>0.001);
        [ indneg, ~ ] = find(delta_pload<-0.001);
        LoadPositiveChange(:,1) = Load(indpos,1);
        LoadPositiveChange(:,2) = delta_pload(indpos);

        LoadNegativeChange(:,1) = Load(indneg,1);
        LoadNegativeChange(:,2) = delta_pload(indneg);

        %% Get the data for the next stage:
        Branch(:,6) = flow_new(:,3);
        Generator(:,2) = Generator_New(:,2);
        Load(:,2) = Load_New(:,2);

        else
        Soln_Flag = 0;
        GeneratorNegativeChange = [];
        GeneratorPositiveChange = [];
        LoadNegativeChange = [];
        LoadPositiveChange = [];
        tot_change_cost = 0;
    end
%% Print the change in dispatches on the screen:
   if (Soln_Flag==1)
        fprintf('------------------------------------------ \n');
        fprintf('Total decrease in load = %f \n',round(sum(LoadNega-
tiveChange(:,2))));
        fprintf('Total increase in load = %f \n',round(sum(LoadPosi-
tiveChange(:,2))));
        fprintf('Total decrease in dispatch  = %f \n', round(sum(Gener-
atorNegativeChange(:,2))));
        fprintf('Total increase in dispatch  = %f \n', round(sum(Gener-
atorPositiveChange(:,2))));
        fprintf('Total change in cost = $ %f
\n',round(tot_change_cost));
        fprintf('------------------------------------------ \n');
   else
        fprintf('------------------------------------------ \n');
        fprintf('No feasible solution obtained! \n');
        fprintf('------------------------------------------ \n');
   end
```

```
    %% Obtain the dc power flow graph:
    NoOfBus = length(Bus);
    Flow_dc = sparse(NoOfBus,NoOfBus);
    Rate_dc = sparse(NoOfBus,NoOfBus);
    for i = 1:length(Branch(:,1))
        Flow_dc(Branch(i,1),Branch(i,2)) = Branch(i,6);
        Flow_dc(Branch(i,2),Branch(i,1)) = (-1)*Branch(i,6);
        Rate_dc(Branch(i,1),Branch(i,2)) = Branch(i,7);
    end
    time = toc;
end
```

```
function [ x] = RoundDown(x)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                                 %
% Program Description: This function rounds down all values                      %
% of a matrix % below 0.02 to zero                                                %
%                                                                                 %
% Author: Reetam Sen Biswas                                                       %
% Arizona State University                                                        %
%                                                                                 %
% Last Modified: 03/20/2020                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    [r,c] = find(abs(x)<0.02);
    for i = 1:length(r)
        x(r(i),c(i)) = 0;
    end
end
```

```
function [ Shortlist, time ] = ShortlistAssets( Branch, EdgeList, Line-
OutNumber )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                       %
% Program Description: This program shortlists the transmission        %
% assets that must be evaluated by the FT following a branch            %
% outage. The logic for this program is based on the SA algorithm       %
%                                                                       %
% Author: Reetam Sen Biswas                                             %
% Arizona State University                                              %
%                                                                       %
% Last Modified: 03/20/2020                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tic;
    [row_renum,col_renum] = size(Branch);
    [row_list, col_list] = size(EdgeList);
    vec = zeros(row_list,1);
    EdgeList = horzcat(EdgeList,vec);
    [row, col] = size(EdgeList);
    count = 1;
    l_Col = [];
    l_Col = find(EdgeList(LineOutNumber,:)==0);
    count = 1;
    Shortlist = [];
    for eno = 1:length(EdgeList(:,1))
        flag = 0;
        e_Col = [];
        e_Col = find(EdgeList(eno,:)==0);
        for e_C = 1:2:e_Col(1)-2
            for l_C = 1:2:l_Col(1)-2
                if (EdgeList(LineOutNumber,l_C)==EdgeList(eno,e_C) &&
EdgeList(LineOutNumber,l_C+1)==EdgeList(eno,e_C+1))
                    Shortlist(count,1) = eno;
                    Shortlist(count,2) = Branch(eno,1);
                    Shortlist(count,3) = Branch(eno,2);
                    count = count+1;
                    flag = 1;
                    break;
                end
                if (EdgeList(LineOutNumber,l_C)==EdgeList(eno,e_C+1) &&
EdgeList(LineOutNumber,l_C+1)==EdgeList(eno,e_C))
                    Shortlist(count,1) = eno;
                    Shortlist(count,2) = Branch(eno,1);
                    Shortlist(count,3) = Branch(eno,2);
                    count = count+1;
                    flag = 1;
                    break;
                end
            end
            if flag==1
                break;
            end
        end
    end
time = toc;
end
```

```
function [ PTDF_true, PTDF_approx, LODF, B_full, H_full, time ] = Up-
date_PTDF_LODF_B_H( B_full, H_full, Bus, Branch, Bran-
chOut,RoundOffFlag)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                       %
% Program Description: This program updates the system matrices         %
% following a branch outage in the system.                              %
%                                                                       %
% Author: Reetam Sen Biswas                                             %
% Arizona State University                                              %
%                                                                       %
% Last Modified: 03/20/2020                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

tic;
%% Updating the H matrix:
    [row_H, col_H] = size(H_full);
    for i = 1:length(BranchOut)
        BranchNum = BranchOut(i);
        F = Branch(BranchNum,1);
        T = Branch(BranchNum,2);
        H_full(BranchNum,:) = zeros(1, col_H);
    end

%% Updating the B matrix:
    % Updating the susceptance (B) matrix changes only four entries of
the
    % matrix and hence saves computation time
    for i = 1:length(BranchOut)
        BranchNum = BranchOut(i);
        F = Branch(BranchNum,1);
        T = Branch(BranchNum,2);
        B_full(F,F) = B_full(F,F)-abs(B_full(F,T));
        B_full(T,T) = B_full(T,T)-abs(B_full(T,F));
        B_full(F,T) = 0;
        B_full(T,F) = 0;
    end

%% Finding the new PTDF matrix:
    noofbus = length(Bus);
    B = B_full([1:noofbus-1],[1:noofbus-1]);
    H = H_full(:,1:noofbus-1);

% Perform matrix operation to obtain the PTDF matrix:
    X = inv(B);
    PTDF = H*X;
    PTDF_true = PTDF;
% For all PTDF values lesser than 0.02, round them down to zero;
    if (RoundOffFlag==1)
       [r,c] = find(abs(PTDF)<0.02);
        for i = 1:length(r)
            PTDF(r(i),c(i)) = 0;
        end
    end
    PTDF_approx = PTDF;
```

```
% From the PTDF matrix, we now create the LODF matrix:
    PTDF_full = horzcat(PTDF,zeros(length(Branch(:,1)),1));
    [nl, nb] = size(PTDF_full);
    f = Branch(:, 1);
    t = Branch(:, 2);
    Cft =  sparse([f; t], [1:nl 1:nl]', [ones(nl, 1); -ones(nl, 1)],
nb, nl);
    H = PTDF_full * Cft;
    h = diag(H, 0);
    LODF = H ./ (ones(nl, nl) - ones(nl, 1) * h');
    h_diff = abs(ones(length(h),1)-h);
    [ pos_ar ] = find(h_diff<0.00001);
    LODF = LODF - diag(diag(LODF)) - eye(nl, nl);
    for  i = 1:length(pos_ar)
       pos_val = pos_ar(i);
       LODF([1:nl],pos_val) = zeros(nl,1);
       LODF(pos_val,[1:nl]) = zeros(1,nl);
       LODF(pos_val,pos_val) = -1;
    end
time = toc;

end
```

```
function [  Flow, Capacity, time ] = UpdateScheme( mpcNewbranch, Line,
Flow, Capacity )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                       %
% Program Description: This program updates the flow and                %
% latent capacity graphs based upon the UPS algorithm. The              %
% logic for this program is based on the UPS algorithm.                 %
%                                                                       %
% Author: Reetam Sen Biswas                                             %
% Arizona State University                                              %
%                                                                       %
% Last Modified: 03/20/2020                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tic;
% Find the flow through the branch
    BusA = mpcNewbranch(Line,1);
    BusB = mpcNewbranch(Line,2);
    NewFlowSheet = Flow;
    NewFlowSheet(NewFlowSheet<0) = 0;
    [Bus1, Bus2, flow] = find(NewFlowSheet);
    found = 0;
    for i = 1:length(Bus1)
        if (Bus1(i)==BusA && Bus2(i)==BusB) || (Bus1(i)==BusB &&
Bus2(i)==BusA)
            FromBus = Bus1(i);
            ToBus = Bus2(i);
            CurrentFlow = flow(i);
            found = 1;
        end
    end
    if found==0
        FromBus = BusA;
        ToBus = BusB;
        CurrentFlow = 0;
    end

% Remove the branch from the flow and latent capacity graphs
    Flow(FromBus, ToBus) = 0;
    Flow(ToBus, FromBus) = 0;
    Capacity(FromBus,ToBus) = 0;
    Capacity(ToBus, FromBus) = 0;

% Re=route the flow through the set of indirect paths:
    FlowCap = 0;
    LoseFlag = 0;
    EdgeTouch = [];
    TouchCount = 1;
    if CurrentFlow==0
        LoseFlag = 1;
    else
        countP = 1;
        while (1<2)
            [S,path]=graphshortestpath(Capacity,FromBus,To-
Bus,'Method','BFS','Directed','true');
```

```
            if S==Inf
                break;
            end
            MaxCap = 9999;
            for k=1:S
                From = path(k);To = path(k+1);
                if MaxCap>Capacity(From,To)
                    MaxCap = Capacity(From,To);
                end
            end

            FlowInj = MaxCap;
            if FlowInj>CurrentFlow
                FlowInj = CurrentFlow;
            end

            for k=1:S
                From = path(k);To = path(k+1);
                Flow(From,To) = Flow(From,To) + FlowInj;
                Flow(To,From) = Flow(To,From) - FlowInj;
                Capacity(From,To) = Capacity(From, To) - FlowInj;
                Capacity(To,From) = Capacity(To, From) + FlowInj;
            end
            countP = countP + 1;

            CurrentFlow = CurrentFlow-FlowInj;
            if CurrentFlow==0
                break;
            end
        end
    end
time = toc;
end
```