Abstract—This paper introduces load shed recovery actions for transmission networks by presenting the dc optimal load shed recovery with transmission switching model (DCOLSR-TS). The model seeks to reduce the amount of load shed, which may result due to transmission line and/or generator contingencies, by modifying the bulk power system topology. Since solving DCOLSR-TS is computationally difficult, the current work also develops a heuristic (MIP-H), which improves the system topology while specifying the required sequence of switching operations. Experimental results on a list of N-1 and N-2 critical contingencies of the IEEE 118-bus test case demonstrate the advantages of utilizing MIP-H for both on-line load shed recovery and recurring contingency-response analysis. This is reinforced by the introduction of a parallelized version of the heuristic (Par-MIP-H), which solves the list of critical contingencies close to 5x faster than MIP-H with 8 cores and up to 14x faster with increased computational resources. The current work also tests MIP-H on a real-life, large-scale network in order to measure the computational performance of this tool on a real-world implementation.

Index Terms—Load Shed Recovery, Contingency Analysis, Transmission Line Switching, Heuristics, Parallel Algorithms.

I. NOMENCLATURE

The following is a list of the mathematical terminology employed in the mathematical model described in this paper (adapted from [1]):

a) Sets:

\( g \in G \) Generators
\( g \in \hat{G} \) Generators out of service due to a contingency
\( k \in K \) Transmission lines
\( k \in \hat{K} \) Transmission lines in service
\( k \in \hat{K} \) Transmission lines out of service
\( k \in K \) Transmission lines out of service due to a contingency
\( n \in N \) Buses

b) Decision variables:

\( P_g \) Power output of generator \( g \)
\( P_k \) Power flow through line \( k \)
\( s_k \) Switch action for line \( k \) (0 - no switch,1 - switch)
\( \theta_n \) Bus angle at bus \( n \)
\( u_n \) Unfulfilled demand at bus \( n \)

This work is supported by the Advanced Research Project Agency - Energy (ARPA-E) under the Green Electricity Network Integration (GENI) program. Adolfo R. Escobedo and Erick Moreno-Centeno are with the Department of Industrial and Systems Engineering at Texas A&M University in College Station, TX, 77843 USA e-mail: (adolfoescobedo@tamu.edu, e.moreno@tamu.edu). Kory W. Hedman is with the School of Electrical, Computer, and Energy Engineering at Arizona State University, Tempe, AZ 85287 USA e-mail: (khedman@asu.edu).

II. INTRODUCTION

FULFILLING the demand for electric power is of utmost importance from economic, protective, and societal standpoints. Nevertheless, utilities are often forced to shed load when there is a significant drop in voltage, insufficient generation, or during other emergency situations. This is done in order to preserve the security and stability of the system, whose increased operational deterioration can result in its absolute collapse. Once demand is shed, system operators and utilities must react jointly to recover it by activating reserves and performing other system-specific measures [2]. Recovery must be swift in order to limit the financial and human costs as interruptions in service continue. Therefore, the methods and tools implemented for load shed recovery (LSR) must be as intelligent and computationally efficient as possible.

Previous work on LSR, also known as system restoration, encompasses post-contingency system normalization procedures as well as black-start schemes. This paper focuses on the former area of study (i.e., system response in the emergency state). Up to this point, the scope of such research has been limited to control actions for distribution networks. Within this purview, the prevailing framework has been distribution feeder reconfiguration via the switching of tie-lines and sectionalizing switches. These topology modifications exploit the radial and loop structures of distribution networks in order to redistribute de-energized loads from isolated out-of-service regions [3]. In addition to finding these improved distribution network topologies, there has been continual development of algorithms and
optimization techniques to specify the feasible sequence of switches involved to achieve them (see Toune et al. [4] for a comparison of these techniques).

This paper introduces LSR control actions for the bulk power system by employing transmission line switching (TS). Hence, it is a study of LSR for transmission networks that works by executing topology changes. The contingencies considered are disturbances on the bulk power network exclusively. Furthermore, the proposed topology reconconfigurations are quite different from those associated with LSR for distribution networks. For starters, the candidates for switching are all the transmission lines in the network. Second, transmission networks have meshed structures, so the effect of lines switches is more intricate than with radial or loop structures. Third, the aim of the topology modifications is to increase the operating capacity of the system beyond its post-contingency capabilities and not to allocate de-energized loads to distribution feeders. Nevertheless, the proposed method can be regarded as an enhancement to current LSR practices. By increasing bulk power system capacity, distribution feeders will have more capacity at their disposal to which de-energized loads may reconnect, thus expediting the LSR process. Having established this connection between the traditional and proposed methods, the remainder of this paper will focus exclusively on LSR for transmission networks.

Previously, TS has been utilized as a corrective mechanism to enhance system security and to reduce line overloads, voltage problems, and line losses [5]. Inclusion into the TS candidate set is often based on experience, whereby a switching scenario is associated with a particular violation [6]. One exception is the algorithm described in Wrubel et al. [7], which ranks switching actions from a pool of user-specified candidates. The algorithm implements a selection strategy that eliminates candidates that do not provide full relief of all pre-switching line overloads; the remaining candidates are ranked or discarded based on their level of post-switching violations measured according to an overload ranking function.

While TS is currently implemented in practice, there do not exist transmission switching software tools to aid operators in the decision-making process. Instead, they simply rely on operating manuals that recommend particular corrective TS actions based on prior experience to alleviate post-contingency violations [8]. For instance, in 2012, PJM took particular high-voltage lines out of service after Superstorm Sandy. Even though the system had already lost multiple high-voltage transmission facilities and implemented involuntary load shedding, additional lines were taken out of service to alleviate overvoltage problems.

In short, while there is a pressing need for better TS tools, the implementation of TS for restorative or corrective purposes has been both restrictive and nonstandardized. On the other hand, beginning with O’Neill et al. [9], the systematic application of TS during normal operations for economic efficiency has been studied extensively through a mixed integer program adaptation of the dc optimal power flow (DCOPF) model. Expanding on this framework, Hedman et al. [1] developed and tested a quick and efficient heuristic that switches one transmission line at a time in order to reduce system generation costs. Moreover, Hedman et al. [10] formulated the problem to satisfy N-1 reliability standards and showed TS can reduce generation costs while obeying these requirements.

The steps involved in the proposed method for load shed recovery with transmission switching are outlined in this paragraph and explained in greater detail in subsequent sections. As a preliminary step of this method, we record the normal state of the system for a given demand level via the DCOPF solution, and we specify the contingency event to consider (when implementing the method on a list of contingencies, recording the normal state is performed only once). We assume the system is in the emergency state from this point, and we disable the system assets corresponding to the contingency. Then, using the normal state dispatch levels (with possible generator ramp down), we solve the associated load curtailment minimization linear model from Pereira and Balu [11] to obtain the difference between demand and fulfilled demand. We record the unfulfilled demand as the load shed associated with the given contingency, but it is more accurately a measure of the potential load shed which may occur (see Section III-B for more details). Finally, we solve the mixed integer dc transmission switching load shed recovery model, DCOLSR-TS (introduced in Section III-A), to obtain the topology which recovers the most load shed.

In practice, the proposed method can be implemented as an on-line or as an off-line application. The on-line application is triggered following a real contingency utilizing actual system status data; its solution provides possible switching actions to the operator in real time. The off-line application serves as a recurring contingency analysis tool that is run 15-min to a day ahead utilizing the unit commitment schedule as the normal state of the system; its solution provides information regarding possible switching actions for a list of critical contingency events. For both of these applications, however, instead of solving DCOLSR-TS directly, we utilize a heuristic (see Section IV-B). This modification is necessary because solving DCOLSR-TS to optimality is computationally difficult and because its solution provides a final topology but not a sequence of switching actions. The heuristic, which switches one line at a time, is practical in both of these aspects. A later section will show the heuristic is effective, efficient, and fast. Moreover, we also parallelize the heuristic and, by doing so, enhance its computational performance.

The remainder of the paper is organized as follows. Section III introduces the new mathematical model and discusses some preliminaries. Section IV outlines related computational issues and describes a heuristic for solving this model. Section V describes the experimental setup for testing the heuristic. Section VI presents the results and analysis. Section VII describes the heuristic parallelization and shows the speedup values it achieved. Section VIII applies the heuristic on a real-life, large-scale network for the purpose of measuring the associated computational performance. Section IX discusses future areas of research and Section X concludes the work.

III. Preliminaries and Mathematical Model

The dc optimal load shed recovery with transmission switching model (DCOLSR-TS) is an adaptation and expansion
from the dc optimal power flow with transmission switching model (DCOPF-TS) from Hedman et al. [1]. First, DCOPF-TS minimizes system generation cost while DCOLSR-TS maximizes load shed recovery. Second, although the (N-1)-compliant DCOPF-TS model integrates emergency state data to guarantee the system can withstand the loss of any generator or transmission line, it is employed during normal conditions [10]. Conversely, DCOLSR-TS operates entirely during emergency conditions (it is a post-contingency corrective action) and, consequently, does not enforce the aforementioned reliability requirements; returning to (N-1) compliance within roughly thirty minutes after the contingency is beyond the model’s scope as well (however, as a later section will show, DCOLSR-TS enhances the system’s ability to approach a state where this process can begin). Third, demand must be completely met in a feasible solution of DCOPF-TS, whereas DCOLSR-TS relaxes this constraint by permitting imbalances between load and generation. The resulting model, minus the TS variables and generator and line status parametrization, is also equivalent to the load curtailment minimization linear model from Pereira and Balu [11].

A. DC Optimal Load Shed Recovery with Transmission Switching (DCOLSR-TS)

DCOLSR-TS is a mixed integer program (MIP). The objective (1) is to maximize the LSR associated with contingency set $\hat{G} \cup \hat{K}$; instructions for the calculation of the parameter $LS_{G\cup K}$ will be detailed in the next subsection. In place of the line status variables of DCOPF-TS, binary switching variables $s_k$ are utilized (12). Constraint (2) limits the difference between bus angles of adjacent buses. While this constraint is frequently not included in DCOPF models since placing a limit on bus angle differences for connected buses is no different than placing a limit on the line’s flow itself, this constraint is included because we also wish to impose a limit on the bus angle differences for lines that are chosen to be switched out of service. Since the lines are likely to be put back into service once the operator tries to regain N-1 reliability, the bus angle differences are kept within this specified bound in order to minimize the impact and stress on the circuit breaker during the reclosure process.

Constraint (3) relaxes the DCOPF node balance constraint by allowing partial demand fulfillment at each node. The imbalance between generation and load is allowed by the introduction of the bounded unmet demand variable $u_n$ (9) at each node $n$. While curtailed load is recovered in lumps in practice, modeling the unmet demand variable as lumpy entails integrating the transmission and distribution models (a distribution network for each sub-transmission interconnection point to a distribution feeder with unfulfilled demand). Since the DCOLSR-TS concept is intended as a support tool to find fast, best-case solutions for enhancing post-contingency LSR capabilities and since it focuses exclusively on the transmission network, modeling the load as lumpy would impose an unjustifiable increase in computational complexity. Hence, $u_n$ is modeled as a continuous variable (see [11] for a similar treatment of this quantity). For the set of lines in service, constraint (4) establishes the lines’ thermal limits, and (5a) and (5b) set their power flow limits based on each line’s susceptance and incident bus angles. Constraints (6)-(7b) are analogous to (4)-(5b) for the set of lines out of service. Constraint (8) sets generation limits for online generators. Lastly, constraints (10) and (11) prevent flow and generation, respectively, for lines and generators out of service due to a contingency.

Subject to:

$$\min \leq \theta_n - \theta_m \leq \max, \quad \forall k(m, n) \in K \quad (2)$$

$$\sum_{k(n, m)} P_k - \sum_{k(n, m)} P_k + \sum_{g(n)} P_g = d_n - u_n, \quad \forall n \in N \quad (3)$$

$$P_{k}^{\min} (1 - s_k) \leq P_k \leq P_{k}^{\max} (1 - s_k), \quad k \in \hat{K} \quad (4)$$

$$B_k (\theta_n - \theta_m) - P_k + s_k M_k \geq 0, \quad k \in \hat{K} \quad (5a)$$

$$B_k (\theta_n - \theta_m) - P_k - s_k M_k \leq 0, \quad k \in \hat{K} \quad (5b)$$

$$P_{k}^{\min} s_k \leq P_k \leq P_{k}^{\max} s_k, \quad k \in \hat{K} \quad (6)$$

$$B_k (\theta_n - \theta_m) - P_k + (1 - s_k) M_k \geq 0, \quad k \in \hat{K} \quad (7a)$$

$$B_k (\theta_n - \theta_m) - P_k - (1 - s_k) M_k \leq 0, \quad k \in \hat{K} \quad (7b)$$

$$P_{g}^{\min} \leq P_g \leq P_{g}^{\max}, \quad g \in G \setminus \hat{G} \quad (8)$$

$$0 \leq u_n \leq d_n, \quad \forall n \in N \quad (9)$$

$$P_k = 0, \quad \forall k \in \hat{K} \quad (10)$$

$$P_g = 0, \quad \forall g \in \hat{G} \quad (11)$$

$$s_k \in \{0, 1\}, \quad \forall k \in K \quad (12)$$

B. Load Shed Calculation

Load shedding is predominantly an automatic system response triggered by relays. Current load shedding practices are divided into under-frequency and under-voltage load shedding, which curtail load based on frequency drop and voltage decline, respectively [12]. The imbalance between generation and demand resulting from generator failures and line faults gradually leads to drops in frequency and voltage. Consequently, line and generator outages can be precursors to load shedding. Based on this causality, we define the amount of load that may be potentially shed by the system as the difference between the normal state system demand and the demand that can be fulfilled following a contingency. Henceforth, this paper will refer to this potential load shed, $LS_{G\cup K}$ (associated with generators $g \in \hat{G}$ and lines $k \in \hat{K}$ out of service due to a contingency), rather than the less tractable load shedding triggered dynamically by the system. Additionally, we will refer to potential load shed simply as load shed (LS) and, therefore, LSR refers to the reduction of the potential load shed within this context.

The LS associated with the contingency denoted by the nonempty set $\hat{G} \cup \hat{K}$ must be obtained prior to solving DCOLSR-TS. For this purpose, the load curtailment minimization linear model from Pereira and Balu [11] is solved with the pre-contingency system status and contingency as inputs.
Equivalently, this model is obtained by modifying DCOLSR-TS as follows. First, the objective (1) is replaced with the expression:

$$\text{Minimize } \sum_{\forall n \in N} u_n$$

(13)

This change is made because the measure of interest is the deficit in demand fulfillment yielded by the generation dispatch at the time the contingency occurs. In a similar vein, the output of each online generator is held as close as possible to its pre-contingency level with the inclusion of the constraint:

$$P_g + P_g^{\text{decrease}} = P_g^{\text{normal}} \quad g \in G \cap \hat{G},$$

(14)

where $$P_g^{\text{normal}}$$ is the generation dispatch during normal conditions and $$P_g^{\text{decrease}}$$ is a nonnegative variable representing the reduction in output of generator $$g$$ from its normal-conditions level. For contingencies involving transmission line faults, $$P_g^{\text{decrease}}$$ must be added because transmission line faults may cause line overloads in other parts of the system [2]. When this occurs, maintaining the exact pre-contingency generator dispatch of online generators may be infeasible. We note that $$P_g^{\text{decrease}}$$ must be included as well for generator-only contingencies since the outage of a generator can overload certain lines previously not overloaded due to a drop in counterflow.

The final modification to DCOLSR-TS to obtain $$LS_{G,K}$$ is disallowing line switching. Notice the contents of $$\hat{K}$$ and $$\hat{G}$$ reflect the normal state topology and those of $$K$$ and $$G$$ represent the contingency set under consideration.

IV. SOLVING DCOLSR-TS

A. Computational Issues

Transmission line switching is in theory an NP-hard problem [1]. In simple terms, this means there are no computationally-tractable methods currently available for solving DCOLSR-TS [13]. Furthermore, the solution of this model yields a modified topology, but it does not specify a feasible sequence of line switches needed to derive it. Such a sequence must be obtained because the violation of any operational constraints must be checked for each change in the topology [14]. This asserts that utilizing the base DCOLSR-TS model as an on-line tool for LSR is both impractical and unjustified.

The above computational complications are associated with obtaining the LSR optimal system topology. In optimization, however, obtaining a good but possibly suboptimal solution in a relatively short amount of time is often the preferred course of action when compared to the implications of solving a problem to optimality. Heuristic algorithms describe the procedures for attaining these favored solutions. In the following subsection we describe a heuristic developed for maximizing load shed recovery with transmission switching.

B. MIP Heuristic

The mixed integer program heuristic algorithm (MIP-H) begins operating as the system enters the emergency state following a contingency. MIP-H finds the best transmission line to switch while considering generation re-dispatch - reachable within a predefined $$\tau$$ minutes of the current state - and then it updates the system accordingly. This process is repeated until the entire system load is fulfilled or until a predefined stopping condition is reached (e.g., maximum number of switching operations, marginal objective value improvement, targeted percentage of $$LS_{G,K}$$ recovered).

In order to obtain the best line switch and generator dispatch, MIP-H solves DCOLSR-TS with the added restriction that only one switch is executed per iteration. This simple modification to DCOLSR-TS yields two significant benefits. First, as will be demonstrated in the next section, it reduces the computational difficulty of solving DCOLSR-TS. Second, it allows for a feasible switching sequence to be derived as the system topology is iteratively improved. A more detailed description of the heuristic follows.

Step 1: heuristic starting point definition

The beginning system state following contingency $$\hat{G} \cup \hat{K}$$ is characterized by $$LS_{G,K}$$ and the corresponding generation dispatch from which it is obtained. For more details on this process, refer to Section III-B. Denote $$P_g^{\text{prev}}$$ as this starting generator output level. Additionally, specify the number of minutes between switching operations, $$\tau$$, and the stopping criteria for the heuristic.

Step 2: generation dispatch limits calculation

Between line switching operations, ramp up/down is allowed for $$\tau$$ minutes. Hence, establish the possible generation levels before the following line switching operation for online generator $$g$$ with the two inequalities:

$$\max\{P_g^{\text{min}}, P_g^{\text{prev}} - \tau r_g\} \leq P_g,$$

(15)

$$\min\{P_g^{\text{max}}, P_g^{\text{prev}} + \tau r_g\} \geq P_g,$$

(16)

Step 3: model definition and solution

Begin with the base DCOLSR-TS model, replace constraint (8) by the simplified inequalities from Step 2, and add the constraint:

$$\sum_{\forall k \in K} s_k = 1$$

(17)

Through these modifications, solving the model equates to finding the best line to switch based on the dispatch achieved within $$\tau$$ minutes of the topology change. Solve the adapted model and save the LSR objective and index of the switched line.

Step 4: system state update

From the Step 3 dispatch solution, update the parameter vector of previous generator output values, $$P_g^{\text{prev}}$$. Then, update the sets of open and closed lines, $$\hat{K}$$ and $$\hat{G}$$, accordingly.

Step 5: stopping condition evaluation

Based on the results obtained in Step 4 and the stopping criteria, determine if the heuristic should perform another iteration. If so, return to Step 2. If not, report the best LSR objective thus far and its associated TS sequence.

V. EXPERIMENTAL SETUP

A. Computing Environment

The experiments were performed on a machine equipped with 32GB of RAM memory and a quad-core 3.6GHz Intel
Table I

<table>
<thead>
<tr>
<th>( P_{\text{min}} ) (MW)</th>
<th>[0,200]</th>
<th>(200, 400]</th>
<th>(400, 600]</th>
<th>(600, ( \infty ])</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r_{g} ) (MW/min)</td>
<td>( P_{\text{min}}/12 )</td>
<td>( P_{\text{min}}/20 )</td>
<td>( P_{\text{min}}/30 )</td>
<td>( P_{\text{min}}/60 )</td>
</tr>
</tbody>
</table>

To specify the contingency instances considered for the test case, the current study applied the concept of a contingency list (CL), which is the collection of valid contingencies periodically examined by contingency analysis programs [16]. Operators often require the consideration of double contingencies, which may cause generation to ramp rates that are unreplicable by separate individual contingency instances [16]. Hence, they were considered as potential elements of CL along with individual contingencies (we consider individual contingencies because the 118-bus test case is N-1 reliable, except for 2 generator units and 3 non-radial transmission lines [10]). However, CL excluded contingencies involving radial lines as in [10]. Altogether, CL considered all single (G-1) and double (G-2) generator failures, all non-radial single (L-1) and double (L-2) line failures, and all mixed generator and non-radial line failures (G-1+L-1) for candidacy.

The study classified each CL candidate as trivial or non-trivial. A contingency is trivial if a re-dispatch - without line switching - recovers all of its associated LS, \( L_{S_{G,K}} \) (we denote the corresponding re-dispatch non-switching model as DCOLSR-nS). Accordingly, nontrivial contingencies are those in which generator re-dispatch does not suffice to recover \( L_{S_{G,K}} \) entirely. In all, out of 3803 CL candidates, there were 863 nontrivial contingencies: 41 generator-only, 427 line-only, and 395 mixed. The proposed DCOLSR-TS method could still be used when generation re-dispatch is sufficient to respond to the contingency (the contingencies labeled as trivial) as it could help reduce the cost of the generation re-dispatch [17]. However, these trivial candidates are discarded since the current analysis focuses on corrective transmission switching applications where generation re-dispatch is not enough, by itself, to ensure no involuntary LS. Hence, CL is composed of the non-radial, nontrivial contingencies.

### C. Heuristic Settings

We set \( \tau = 10 \) (minutes between switching operations), and defined the stopping criteria of the heuristic as follows. MIP-H finds the next line to switch until all of \( L_{S_{G,K}} \) is recovered or until the current iteration LSR represents less than a 0.01% improvement over the previous iteration LSR. We also adopted a single ramp rate \( r_{g} \) per generator, as constructed above in Table I, for simplification purposes. Furthermore, we assumed all generators are committed, and the system load remains equal to the base load during algorithm execution.

### VI. RESULTS AND ANALYSIS

Table II displays solution metrics evaluated to compare LSR without TS to LSR with TS (labeled DCOLSR-nS and DCOLSR-TS on the table, respectively) and to compare the two techniques for LSR with TS: MIP-H and solving DCOLSR-TS directly (CPLEX). We note that the CPLEX solutions may not be optimal based on the 1 hr solver time limit specified in Section V-A; when the time limit is exceeded the best feasible solution is reported. Table II gives load shed recovery percentages (LSR%), optimality gap percentages (Gap%), number of TS operations (Switches), and computational times in seconds (Time). LSR% is obtained by dividing LSR by \( L_{S_{G,K}} \) and multiplying by 100. Gap% is calculated as follows:

\[
\text{Gap\%} = \frac{\text{Upper Bound} - \text{Lower Bound}}{\text{Upper Bound}} \times 100
\]

where the lower bound is the best feasible solution found within the 1 hr time limit, and the upper bound is the best upper bound reported by CPLEX when solving DCOLSR-TS directly with the 1 hr time limit enforced as well.

### TABLE II

<table>
<thead>
<tr>
<th>CL-ALL</th>
<th>DCOLSR-nS</th>
<th>DCOLSR-TS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MIP-H</td>
<td>CPLEX</td>
</tr>
<tr>
<td>AVG</td>
<td>AVG</td>
<td>AVG</td>
</tr>
<tr>
<td>SD</td>
<td>SD</td>
<td>SD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LSR%</th>
<th>Gap%</th>
<th>Switches</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>54.50</td>
<td>24.88</td>
<td>28.32</td>
<td>28.35</td>
</tr>
<tr>
<td>60.00</td>
<td>0.58</td>
<td>2.09</td>
<td>0.10</td>
</tr>
<tr>
<td>10.99</td>
<td>3.63</td>
<td>2.41</td>
<td>10.99</td>
</tr>
</tbody>
</table>

The table gives results for the whole contingency list (CL-ALL) and for the three subgroups of CL by contingency type: generator-only (G-1&G-2), line-only (L-1&L-2) and mixed (G-1+L-1). Specifically, the results are displayed in per-contingency average (AVG) and standard deviation (SD) values for each of the four table divisions discussed. The following subsections highlight findings associated with this table and with additional experiments.
A. Load Shed Recovery with Transmission Switching vs Load Shed Recovery without Transmission Switching

Several key observations arise from Table II. To begin with, the benefit of TS for LSR is readily identified by comparing the LSR% average values; these are also highlighted graphically by Fig. 1. The DCOLSR-TS LSR% average (CPLEX column) is over 33 percentage points greater than the non-switching LSR% average (DCOLSR-nS column) for CL-All. When CL is subdivided by contingency type, the percentage point difference in these LSR% subgroup average values is at least 23.64 and as high as 41.97; the larger difference corresponds to a TS LSR% more than twice than without TS for line-only contingencies. The significant increase in LSR% from DCOLSR-nS to CPLEX indicates, in effect, that the capacity of the system can be significantly expanded with TS. Therefore, TS allows the system to reach an operating point where less load is curtailed while fault repair and other recovery measures are implemented, thereby expediting system normalization and the return to N-1 reliability.

B. MIP-H vs CPLEX

The CPLEX solution has two drawbacks: excessive average number of TS operations and high average computational times. MIP-H alleviates both concerns overwhelmingly. The average number of TS operations for CL-All with MIP-H is less than a sixth of the CPLEX average. More impressively, with MIP-H, L-1&L-2 contingencies yield only 1.01 switching operations, whereas CPLEX yields an excess of 15, on average. This reduced number of TS operations renders MIP-H more practical than CPLEX since the number of LSR topology changes should be as small as possible during implementation [14]. Further experiments will show even more practicality in this regard.

This section presents Fig. 2, a Dolan and Moré [18] performance profile, to compare the computational times of MIP-H and CPLEX. Specifically, for a sequence of target times, the featured performance profile measures the % of CL elements for which each LSR technique achieved a computational time of at most each target time; the target times are positive multiples of the best computational time achieved between MIP-H and CPLEX for each CL element.

As the graph shows, MIP-H dominates CPLEX computationally since, for every target time, MIP-H achieved a computational time of at most the target time for a larger % of CL elements than CPLEX. In the worst case, an instance of MIP-H is 15x slower than CPLEX, while an instance of CPLEX can be 2047x slower than MIP-H. Moreover, the maximum times recorded for MIP-H and CPLEX were 9.56s and 3600s, respectively (for CPLEX, 3600s is actually a lower bound since we set the maximum solver time to 1 hr). Unlike the method of solving DCOLSR-TS directly with CPLEX, therefore, MIP-H offers the computational consistency ideal for implementation.

Besides being efficient and practical, MIP-H is effective. Remarkably, as shown by Table II, its average gap for CL-All is 0.06% with a standard deviation of 0.58%, which is only 0.04 and 0.45 percentage points higher than corresponding measures for CPLEX (a nonzero CPLEX Gap% results when the optimal solution is not obtained within the one-hour time limit). In addition, when CL is subdivided by contingency type, the MIP-H average gap is always well below 1%, and MIP-H achieves optimality for all L-1&L-2 contingencies. Fig. 1 accentuates the insignificant differences in LSR% magnitudes between the average MIP-H and CPLEX solutions. Hence, the observations presented thus far show MIP-H sacrifices little in solution quality in exchange for dramatically superior computational performance and practicality. Moreover, since line-only contingencies are generally more common than generator-only or mixed contingencies (49.48% of CL), their exceptional results can be expected to occur frequently in other networks.

C. MIP-H vs traditional generation re-dispatch

This subsection demonstrates the advantage of implementing TS for LSR by showing the system recovers LS at a quicker pace than when TS is not included as a control action. For this purpose, we compare the LSR% given by MIP-H and the re-dispatch non-switching model, DCOLSR-nS, for CL at three post-contingency points: 10 minutes (10min), 20 minutes (20min), and at the conclusion of each process (Final). The associated LSR values for DCOLSR-nS are obtained by

![Fig. 2. Performance Profile - MIP-H vs CPLEX](image-url)
restricting each generator’s output based on its initial post-contingency output, time allotted to ramp, and its ramp rate. The results are plotted on Fig. 3.

![Comparison of MIP-H and DCOLS-nS](image)

Fig. 3. Comparison of MIP-H and DCOLS-nS

A salient finding is MIP-H has recovered strictly more of $LS_{G,K}$ than DCOLS-nS Final for over 90% of CL after one switching operation (10min). After only two switching operations (20min), MIP-H recovers at least 80% of $LS_{G,K}$ for 88.18% of CL instances, which is only 1.85% less CL instances than MIP-H Final. This shows MIP-H is able to reap almost all of its benefits within twenty minutes or two line switches, which is critical since this allows the system to prepare for another possible contingency faster. Moreover, MIP-H recovers 100% of $LS_{G,K}$ for 23.99% and 42.76% of CL instances after 10 minutes and 20 minutes, respectively. For these contingencies, consequently, MIP-H averts the need for other system recovery measures within a short period of time. Therefore, in a real implementation of MIP-H, another stopping condition limiting the number of switches can be added, thereby increasing the practicality of the method while retaining the potential for very favorable results.

VII. MIP-H PARALLELIZATION

A. Algorithm Design

High-performance computing can enhance MIP-H’s computational performance. To demonstrate this, the current section describes a message-passing (MP) parallel algorithm for LSR (Par-MIP-H). In an MP algorithm, $p$ processing cores perform their processes independently and only have access to their own memory. In order to share data among cores, messages are passed explicitly. Par-MIP-H works by partitioning the set of transmission switching decision variables of DCOLS-TS among $p$ cores (labeled individually as $\rho_0, \rho_1, ..., \rho_{p-1}$). The partition is such that the number of these binary variables per core is as even as possible. Hence, given a specific assignment rule, a network of $|K|$ transmission lines, and $p$ cores (where $1 \leq p \leq |K|$), Par-MIP-H assigns $\lfloor \frac{|K|}{p} \rfloor$ and $\lfloor \frac{|K|}{p} \rfloor$ distinct TS variables to each of $|K| \mod p$ and $p - |K| \mod p$ cores, respectively. The unique subset of TS variables of $\rho_i$ is denoted as $K_{\rho_i}$, where $i = 0, 1, ..., p - 1$.

Par-MPI-H follows MIP-H’s steps with some minor modifications (refer to Section IV-B for the description of MIP-H). As a preliminary step, however, each core determines its unique subset of TS variables based on a pre-specified assignment rule. Then, $\rho_0$ hereafter referred to as the master core, executes Step 1 (heuristic starting point definition) and broadcasts the network and contingency data to all other cores. Next, every core performs Step 2 (generation dispatch limits calculation) and Step 3 (model definition and solution) with the distinction that in addition to equation (17), $\rho_i$ includes the constraint:

$$\sum_{k \in K_{\rho_i}} s_k = 1$$

(19)

The two constraints (17 and 19) together imply $\rho_i$ finds its switched line from $K_{\rho_i}$, and it prevents switching for all other lines. After including these constraints and solving the resulting model, each core sends its objective value to the master core. The master core then determines the maximum of these LSR values and communicates with the core who achieved the maximum in order to receive the associated switched-line index and generator dispatch. Subsequently, the master core performs Step 4 (system state update) by broadcasting the switched-line index and generator dispatch. Lastly, the master core evaluates the stopping condition and instructs all other cores to return to Step 2 to begin a new iteration or to stop (as in Step 5 of MIP-H).

B. Computing Environment

We performed the parallel experiments utilizing a maximum of 23 multi-processor Nehalem-type computing nodes connected via a 4X QDR Infiniband. Nehalem is a registered trademark of Intel, Inc. Each of these nodes is equipped with 22GB of RAM memory shared among 2 2.8GHz quad-core Intel Xeon 5560 processors. The operating system and software were the same as those detailed in Section V-A with the addition of the OpenMPI 1.4.3 library.

C. Experiments and Results

Fig. 4 displays Par-MIP-H’s computational performance in the form of average speedup values. Speedup is defined as the ratio $\frac{T_s}{T_p}$, where $T_s$ and $T_p$ are the computational times to solve a problem instance by a sequential algorithm (MIP-H) and by a parallel algorithm (Par-MIP-H) utilizing $p$ cores, respectively. In this case, the problem instance is the whole CL described in Section V-B. On the graph, the data points’ $x$-coordinates give 21 distinct values of $p$, and their $y$-coordinates list the corresponding average speedup values calculated over 10 repetitions. We selected these values for $p$ and number of repetitions to provide a representative speedup curve while adhering to computational resource constraints. Moreover, the standard deviation - at most .03 for 20 out of 21 data points and .12 for the remaining point ($p = 186$) - is not shown because of its relatively insignificant magnitude and to provide visual clarity.
At first, it appears the Par-MIP-H speedup curve follows the typical pattern of parallel algorithms: as \( p \) increases, speedup rises fastest for smaller values of \( p \) and eventually plateaus. Indeed, from \( p = 1 \) to \( p = 8 \), \( p = 16 \) to \( p = 32 \), and \( p = 56 \) to \( p = 152 \), the speedup values increase, but less so as \( p \) becomes larger. Unlike typical parallel algorithms, however, the slope of Par-MIP-H speedup curve shows an estimable increase just as it begins to stagnate at \( p = 152 \). In fact, with an increment of 106 less cores, the net speedup from \( p = 152 \) to \( p = 186 \) is .9 greater than from \( p = 12 \) to \( p = 152 \). This unexpected boost in computational performance is impressive but, since it requires one core for every transmission line (numbering in the thousands in typical power systems), it may be too unreasonable and expensive to achieve. Nevertheless, Par-MIP-H reduced MIP-H’s computational time for solving the whole CL of 20.93min to 3.78min and 1.44min, on average, utilizing 8 and 186 cores, respectively (we remark that the discrepancy between the sequential running times of 20.93 min and 13.52 min of this section and Section VI, respectively, is attributed to the differences in computing environments). Thus, Par-MIP-H achieved over a third of the maximum speedup, or 88% of the maximum computational time reduction, with the number of cores equipped in many of today’s desktop computers.

The abnormal speedup curve behavior in the right-most segment of Fig. 4 is largely attributable to the structure of the particular models solved by each core. In particular, for \( p \geq 93 \), 186 – \( p \) cores solve mixed integer programs and 2\( p \) – 186 solve linear programs. Thus, for these values of \( p \), the number of cores solving linear programs rises along with \( p \). Additionally, for the two speedup decreases at \( p = 12 \) and \( p = 40 \), Par-MIP-H assigns at least half of the cores with 1 more binary variable than the remaining cores, and this causes a bottleneck effect. Another possibility for these speedup dips is an imbalance in computational difficulty created by some TS subset assignments. This experiment allocated these subsets sequentially with respect to increasing line indices and decreasing subset sizes. It is possible other assignments could produce different results. Hence, to exploit potential speedup fully, an implementation of Par-MIP-H should include preliminary testing of different TS subset assignments based on the size and structure of the network and the computing resources available.

**VIII. LARGE-SCALE TESTS**

This section measures the computational performance of MIP-H on a real-life, large-scale network released by the Federal Energy Regulatory Commission (FERC) of the USA. The scope of this analysis focuses on the timing aspects of a realistic implementation of this tool.

**A. Test Case Specification**

The test case consists of close to 13k buses, 1k generators, and 19k transmission lines. There are 306 generators marked as committed, and their capacities sum to 76GW. The experiment will dispatch only these 306 generators and set their dispatch lower limits to 0 since DCOLSR-TS does not make unit commitment decisions (as in [1]). The base load, totaling 68GW, corresponds to the hour with highest system load that can be served in the normal state by the committed generation units. Furthermore, the linear cost for each generator was calculated by taking a weighted average of its cost intervals (this cost is utilized only to derive the pre-contingency state via the DCOPF solution).

**B. Experiments and Results**

We tested MIP-H on 30 contingency events divided equally among double generator (G-2), double line (L-2), and mixed generator and line (G-1+L-1) failures. Table 3 displays the average number of MIP-H iterations (i.e., number of line switches followed by 10min ramp periods), the average computational time per MIP-H iteration (in minutes), and the corresponding standard deviations of these averages. The results are shown for the 30 contingencies as a whole (All) and for each of the above contingency subgroups.

**TABLE III**

<table>
<thead>
<tr>
<th>MIP-H LARGE-SCALE COMPUTATIONAL PERFORMANCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterations AVG</td>
</tr>
<tr>
<td>-------------------------------------------</td>
</tr>
<tr>
<td>Iterations SD</td>
</tr>
<tr>
<td>Time/Iteration AVG (min)</td>
</tr>
<tr>
<td>Time/Iteration SD (min)</td>
</tr>
</tbody>
</table>

The number of MIP-H switches is comparable to those on Table II, but the computational times are higher. However, considering the large-scale test case consists of over 10x more decision variables and over 100x more constraints than the IEEE 118-bus test case, MIP-H scales reasonably well. In fact, a computational time of 1min per contingency, which seems reasonable for real-world implementation, may soon be within reach. This is because the computational capability of modern optimization software has increased by a factor greater than 1000 in less than ten years, and experts believe this trend will continue [19]. Additionally, this improvement factor
is independent of expected advances in computer hardware. Hence, based on these observations and the potential speedups of Par-MIP-H, we contend large-scale test cases will be solved fast enough for on-line implementation and 15-min to 1h ahead recurring contingency-response analysis within the coming few years. In the meantime, MIP-H and Par-MIP-H can be utilized in an off-line capacity to derive possible topology control actions for dealing with serious contingencies under varying system demand levels.

IX. Future Work

This research is funded by the Advanced Research Project Agency - Energy (ARPA-E) under the Green Electricity Network Integration (GENI) program. Ongoing and future work includes developing a real-time tool that incorporates the proposed optimization algorithm along with AC feasibility and stability checks to confirm the capability to implement the proposed switching action. Additional information can be found on the project’s website (http://smartgridcenter.tamu.edu/ratc).

X. Conclusion

This paper introduced a static transmission switching model for load shed recovery (DCOLSR-TS). Tests of DCOLSR-TS and its non-switching counterpart utilizing the IEEE 118-bus test case showed the advantage of implementing transmission switching following critical N-1 and N-2 line and/or generator contingencies for reducing the amount of load that may be curtailed by the system. Specifically, transmission switching reduced this quantity by an average of 33 more percentage points than the re-dispatch non-switching model. In essence, TS is able to significantly expand the emergency capacity of the system, enabling it to begin the normalization process and the return to N-1 reliability quicker.

The current work also described a heuristic for solving DCOLSR-TS (MIP-H), which achieved average optimality gaps of less than 1% while switching a significantly fewer number of lines. Furthermore, MIP-H reduced the computational difficulty of DCOLSR-TS appreciably. In particular, its average computational time per contingency analyzed was 1.01% of that of solving DCOLSR-TS directly with CPLEX. Other tests showed MIP-H attained close to all of its potential benefits within the first two switching operations following the contingency event. Therefore, MIP-H is practical, effective, and efficient. The paper presented a parallelization of MIP-H as well (Par-MIP-H). Its average computational time per contingency analyzed was 18.05% of MIP-H's average utilizing only 8 cores and as fast as 6.86% of MIP-H's average with 186 cores (i.e., using one core per line).

Finally, this work applied MIP-H to a real-life, large-scale test case to measure its computational performance. Although the computational times were higher than in the 118-bus test case, the target computational times for implementation are reachable within a few years based on expected advancements in computer hardware and optimization software and on the observed parallelization speedups. Thus, we expect MIP-H soon will be sufficiently fast for both on-line load shed recovery and recurring contingency-response analysis. In the meantime, however, it can still be used as a powerful off-line contingency analysis tool.

ACKNOWLEDGMENT

The authors acknowledge the Texas A&M Supercomputing Facility (http://sc.tamu.edu/) for providing computing resources useful in conducting the research reported in this paper.

REFERENCES

Adolfo R. Escobedo (S’13) received the B.A. degree in mathematics from California State University, Los Angeles, in 2009. Currently, he is pursuing the Ph.D. degree in industrial and systems engineering from Texas A&M University. His research interests include optimization and power systems operations and planning.

Erick Moreno-Centeno received the B.S. degree in industrial physics engineering from the Instituto Tecnológico y de Estudios Superiores de Monterrey (ITESM), Mexico in 2002 and the M.S. in computer science from the University of California, Berkeley in 2010. He received the M.S. and Ph.D. in industrial engineering and operations research from the University of California, Berkeley in 2006 and 2010, respectively. Currently, he is an assistant professor in the Department of Industrial and Systems Engineering at Texas A&M University. His research interests are in the general area of mathematical programming, particularly in the design and analysis of combinatorial optimization models and algorithms to solve complex problems.

Kory W. Hedman (S’05, M’10) received the B.S. degree in electrical engineering and the B.S. degree in economics from the University of Washington, Seattle, in 2004 and the M.S. degree in economics and the M.S. degree in electrical engineering from Iowa State University, Ames, in 2006 and 2007, respectively. He received the M.S. and Ph.D. degrees in industrial engineering and operations research from the University of California, Berkeley in 2008 and 2010, respectively. Currently, he is an assistant professor in the School of Electrical, Computer, and Energy Engineering at Arizona State University. He previously worked for the California ISO (CAISO), Folsom, CA, on transmission planning and he has worked with the Federal Energy Regulatory Commission (FERC), Washington, DC, on transmission switching.