Computational Study of Security Constrained Economic Dispatch with Multi-stage Rescheduling

Yanchao Liu, Michael Ferris and Feng Zhao

Abstract—We model post-contingency corrective actions in the security-constrained economic dispatch and consider multiple stages of rescheduling to meet different security constraints. The resulting linear program is not solvable by traditional LP methods due to its large size. We devise and implement a series of algorithmic enhancements based on the Benders’ decomposition method to ameliorate the computational difficulty. These enhancements include reducing the number of subproblems, solving the LPs faster by using appropriate solver options, harnessing parallel computing and treating difficult contingencies separately by integrating an independent “feasibility checker” process in the algorithm. In addition, we propose a set of online measures to diagnose and correct infeasibility issues encountered in the solution process. The overall solution approach, coded directly in GAMS, is able to process the “N-1” contingency list in ten minutes for all large network cases (e.g., the Polish 2383-bus case) available for experiments.

I. INTRODUCTION

A. Motivation

ECONOMIC efficiency and system reliability are top concerns in the day-to-day operation of the restructured energy market over the grid. However, the two goals are intrinsically competing with each other. Efficiency pushes for the maximum use of available transmission capacity to facilitate the merit-order dispatch of generation resources, while reliability requires a certain degree of conservatism in the use of transmission capacity to prepare for unexpected events such as line and generator outages. To balance the two goals, system operators typically solve a security-constrained economic dispatch (SCED) model. Economic dispatch (ED) seeks a nodal injection/withdrawal arrangement (i.e., dispatch solution) to minimize the total generation cost in a base-case network setting. Security constraints (SC) require that the economic dispatch solution must simultaneously support a feasible power flow under a list of counterfactual scenarios of component failure, called contingencies. A feasible power flow is one that does not cause overloads in lines, as power flow automatically redistributes across lines following physical laws in case of a contingency. When the contingency list spans all elements in the system, the corresponding SCED solution, if one exists, is said to meet the N-1 security criterion. In practice, the solution is oftentimes obtained or approximated via an iterative process: obtain an ED solution and test if it is feasible for all contingency cases, if not, refine the solution and test again. This process is termed as a simultaneous feasibility test (SFT).

Practical reliability standards usually allow for some flexibility in the security constraints. In particular, the post-contingency power flow may temporarily exceed the normal line rating as long as system operators are able to correct it in a limited amount of time via rescheduling actions. For example, ISO New England uses four levels of thermal capacity ratings for transmission facilities: Normal, Long Time Emergency (LTE), Short Time Emergency (STE) and Drastic Action Limit (DAL), with increasing rating numbers. While the transmission line and equipment loadings should not exceed the Normal rating for pre-contingency system conditions, the operating procedure (OP) [1] approves the use of other (less restrictive) ratings under contingency conditions. Specifically, the OP requires that the post-contingency line loadings should not rise beyond the DAL and must be reduced below the STE rating in 5 minutes, reduced below the LTE rating in 15 minutes and reduced below the Normal rating in 30 minutes (see Figure 1 for an illustration). Although imposing the Normal rating at all times is sufficient for reliability, the relaxed standards should be properly implemented in the SCED software to preserve economic efficiency. However, there is no evidence that the post-contingency rescheduling procedures are actually considered in prevalent dispatch software. We postulate that the main obstacle comes from the computational difficulty due to the increased model size.

In this paper, we present a SCED model that takes the multi-stage contingency response actions into account. In order to solve large instances of the model, we develop a series of algorithmic enhancements based on the Benders’ decomposition method. We also analyze the causes of infeasibility and propose an approach to diagnose and correct infeasible situations in the solution process. Thus our solution approach provides not only an optimal dispatch solution, but also a list of contingencies that need to be treated separately.

B. Related Work

SCED with corrective rescheduling (SCED-C) has been studied for over two decades. The pioneering work by Monticelli et al. [2] described the mathematical framework with great clarity and illustrated the economic gain of taking into account system rescheduling capabilities. The authors also pointed out many extensions that motivated our work, including multiple dispatch steps each considering different
line ratings for different time frames of emergency control, and the prospect of processing the subproblems in parallel. Recent advances have been made along two major avenues: (1) contingency filtering (CF) techniques to effectively reduce the problem size, e.g., [3]–[5] and (2) decomposition and parallel algorithms to obtain/approximate global solutions efficiently, e.g., [6], [7].

Capitanescu and Wehenkel [3] studied the (single-stage) corrective security-constrained AC optimal power flow (CSCOPF) problem. The authors exploited the fact that in practice most contingencies are not binding at the optimum by iteratively solving CSCOPF (using an interior-point method [8]) with increasing size. In each iteration, the CSCOPF model only incorporates those post-contingency constraints that have been identified to be “potentially binding” by a contingency filtering procedure. In another paper, i.e., [4], Capitanescu et al. proposed two CF techniques to efficiently identify a minimal “dominating” subset of contingencies, the complement set of which is redundant for the solution of SCOPF and can be removed, thus reducing the size of the problem. A recent work by Fliscounakis et al. [5] incorporates uncertain demand in the SCED-C context and uses a mixed integer bi-level optimization model to ensure a worst-case coverage of the dispatch solution. The authors ranked contingencies into four clusters based on severity and carefully chose the solver options for computational performance. In the present work, we embed a contingency filtering idea in the Benders’ algorithm. Compared to existing work in the literature, our method has three desirable features. First, the CF step does not incur extra computation load. Second, the filtering is not a once-and-for-all procedure, but is dynamically integrated in the iterative algorithm. Third, the filtering requires minimal domain-knowledge-based judgement about the network or contingency but is entirely based upon numerical results of the subproblem. Admittedly, domain knowledge might also be useful to augment our method.

To solve the nonlinear nonconvex SCOPF problem, Phan and Kalagnanam [6] investigated a global optimization algorithm based on Lagrangian duality, as well as two decomposition schemes, namely, Benders’ decomposition and the alternating direction method of multipliers (ADMM). Since a Benders’ cut is not valid (i.e., may cut off feasible regions and the global solution) in the nonconvex AC context, as a computational alleviation the authors proposed to shift the cutting plane by an adaptively chosen distance so as to cut off less of the feasible region. The authors also briefly remarked that a contingency, once feasible for the base-case solution, can be “switched off” from future iterations. We derive a concrete and rigorous algorithmic enhancement that includes adaptive switching off of contingencies and demonstrate its effectiveness quantitatively. Pinto and Stott [9] briefly reviewed the Benders’ algorithmic framework applied to SCED-C, and stressed that a computational study on a full-scale prototype was needed, which we aim to provide in this paper.

Parallel computing is becoming a standard technique for “large-scale” computation in decomposable systems. Recent work from Argonne National Laboratory, i.e., [7], [10], provided efficient parallel algorithms for solving huge LPs. The algorithms were based on interior-point methods and a Schur complement technique, which the authors demonstrated to achieve a high scaling efficiency on supercomputers. In this paper, we implemented the well-established scheme of parallel computing for Benders’ decomposition, with the aim of showcasing its practical effectiveness on an affordable computing server, hence providing a realistic estimate of deployment potential of our model.

In the aspect of modeling, Capitanescu and Wehenkel [11] warned that if the immediate post-contingency state (power flow) violates limits too much, the system may collapse before corrective actions take effect. They postulated a constraint to be added to the corrective SCED model in order to prevent this from happening. Our multi-stage corrective model naturally contains such a constraint, i.e., the one for period $T = 0$ with the DAL line rating. While part of the SCED-C literature is based on the AC power flow equations, e.g., [2] and [3], in the present work a linear “DC” model is more aligned with our objective. First, the decomposition theory for convex optimization is well-established and proven to guarantee global solutions, so we can focus on algorithmic enhancements for faster solution. Second, most ISO/RTOs use a linear model in the dispatch software, thus our algorithm as well as the computational results can directly compare with those of the existing software, enabling more credible evaluation of its industrial potential. Readers can consult [12] for an insightful review of the SCOPF problem and methodology.

II. THE MODEL AND ITS STRUCTURE

SCED with post-contingency corrective rescheduling (SCED-C) is written in the general form (with $K$ contingencies) as follows [2], [3], [9]:

$$
\begin{align*}
\min_{x_0, \ldots, x_K, u_0, \ldots, u_K} & \quad f_0(x_0, u_0) \\
\text{s.t.} & \quad g_k(x_k, u_k) = 0 \quad k = 0, \ldots, K \\
& \quad h_k(x_k, u_k) \leq 0 \quad k = 0, \ldots, K \\
& \quad |u_k - u_0| \leq \Delta_k \quad k = 1, \ldots, K
\end{align*}
$$

(1)

where $f_0$ is the base-case objective function and $h_k$ and $g_k$ are constraint functions. For the $k$-th system configuration, $x_k$
is the vector of state variables and $u_k$ is the vector of control variables. $\Delta_k$ is the vector of maximal allowed variation of control variables between the base case ($k = 0$) and the $k$-th post-contingency configuration.

In a simplified linear “DC” network setting which we work with in this paper, the control variables are the generation level $P$ and the state variables are the voltage angle $\delta$ and the line flow $F$. The equality $g(x,u) = 0$ corresponds to

$$\sum_{g(i)} P_g - \sum_{(j,c) \in BR} F_{i,j,c} + \sum_{(j,c) \in BR} F_{j,i,c} = D_i \quad \forall i \in BUS$$

and the inequality $h(x,u) \leq 0$ corresponds to

$$F_{i,j,c} - F_{i,j,c}^* \leq 0 \quad \forall (i,j,c) \in BR$$

and $F_{j,i,c} - F_{j,i,c}^* \leq 0 \quad \forall (i,j,c) \in BR$ (2)

$$P_{g_{\min}} - P_g \leq 0 \quad \forall g \in GEN$$

(4)

$$P_{g_{\max}} - P_g \leq 0 \quad \forall g \in GEN$$

(5)

For simplicity we consider a linear cost function, i.e., $f_0(x_0, u_0) := c_0^T u_0$, where $c_0$ can be regarded as the marginal cost of generation. There may be multiple lines (or circuits) between two buses, hence the branch $(i,j,c)$ indicates the $c$-th circuit connecting bus $i$ and $j$. Functions $g$ and $h$ are identified by the following sets and parameters:

<table>
<thead>
<tr>
<th>BR, BUS, GEN</th>
<th>Set of branches, buses and generators</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_i$</td>
<td>Fixed load at bus $i$</td>
</tr>
<tr>
<td>$g(i)$</td>
<td>Set of generators connected at bus $i$</td>
</tr>
<tr>
<td>$b_{i,j,c}$</td>
<td>Susceptance of branch $(i,j,c)$</td>
</tr>
<tr>
<td>$F_{i,j,c}$</td>
<td>Thermal rating of line $(i,j,c)$</td>
</tr>
<tr>
<td>$p_{\min}$, $p_{\max}$</td>
<td>Bounds of generator output</td>
</tr>
</tbody>
</table>

In this paper, contingencies are the single line outages and there are several levels of post-contingency line ratings, so the functions $g_k$ and $h_k$ of post-contingency configuration $k$ is actually represented by the set $BR_k$ and parameters ($F_{i,j,c})_k$. The feasible region of a DC-based SCED is the intersection of many polyhedra, as illustrated in Figure 2.

It is well-known that the linear DCOPF model is a coarse representation of the physics of AC circuits in power systems. In particular, the power flow equations outlined above ignore real power losses as well as reactive power constraints. In fact, it is acceptable practice not to consider the reactive power in the economic dispatch solution which is primarily used to settle the market for real power. Detailed AC power flow studies usually follow at a later stage where various ancillary services come into play. It is possible to incorporate system losses in the linearized DC formulation, see [13]. However, because line loss is a quadratic function of the real power flow, it takes multiple iterations of the DCOPF run to achieve an accurate approximation of the total system loss as well as the loss factors (LF), which determine how the total losses are distributed/compensated across buses. Due to computational constraints, the ISOs often solve the real-time dispatch problem with estimated system loss and LFs and without the iterative process. This can be easily incorporated into our model. Therefore, our choice of a DC model is practical.

We have tested an extension of the model to account for line losses in the base case, adopting the fictitious nodal demand (FND) idea from [13] (basically, evenly dividing/allocating the estimated loss on line $(i,j,c)$ to the buses $i$ and $j$ as FNDs). The computational performance is indistinguishable to that of the model without considering the losses. For simplicity and limited by data availability, we use the lossless formulation for subsequent discussion and experiments.

### A. SCED with Multi-stage Rescheduling

We now extend the general formulation (1) to the multi-stage rescheduling situation, where the time dimension plays an explicit role. The time index $t$ will appear as superscripts on applicable symbols and the subscript $k$ now indexes contingency cases with $k = 0$ being the pre-contingency case (base-case). Suppose the post-contingency operating procedure involves $T$ checkpoints in time and there are $K$ contingencies to prepare for in the SCED. When the system is operating at normal state $(x_0, u_0)$, base-case feasibility requires that $g_0(x_0, u_0) = 0$ and $h_0(x_0, u_0) \leq 0$. When contingency $k$ occurs, the state variable $x$ will instantaneously change to $x^*_k$ following physical laws, i.e., $g_k(x^*_k, u^*_k) = 0$, where $u^*_k = u_0$ since the control variable cannot change abruptly. In general, security constraints require that the following conditions hold

$$g_k(x^*_k, u^*_k) = 0$$

$$h_k(x^*_k, u^*_k) \leq \epsilon_t$$

$$|u^*_k - u_0| \leq \Delta_t$$

for a discrete set of time checkpoints $t$. For instance, ISO New England’s operating procedure imposes the following checkpoints:

- $t = 0$ corresponds to the immediate checkpoint to ensure that the line flow is within the STE rating, the components of $\epsilon_t$ for (4) and (5) are 0 (same for cases below) and the components of $\epsilon_t$ for (2) and (3) is $F^{DAL} - F^{Normal}$ (for notation convenience, we write $\epsilon_t = F^{DAL} - F^{Normal}$, $\Delta_t = 0$).
- $t = 1$ corresponds to the 5-minute checkpoint to ensure that the line flow is reduced within the STE rating, $\epsilon_t = \ldots$
\[ F_{\text{STB}} - F_{\text{Normal}}, \Delta_t = 5R, \] where \( R \) is the vector of per minute ramp rate of injection at the buses.

- \( t = 2 \) corresponds to the 15-minute checkpoint to ensure that the line flow is reduced within the LTE rating, \( \epsilon_t = F_{\text{LTE}} - F_{\text{Normal}}, \Delta_t = 15R. \)
- \( t = 3 \) corresponds to the 30-minute checkpoint to ensure that the line flow is reduced within the Normal rating, \( \epsilon_t = 0, \Delta_t = 30R. \)

For a given contingency \( k \), condition (6) requires that there exists a recourse solution \( u_k^0 \) for the time period between when the contingency occurs and the time of the checkpoint \( t \). However, satisfying (6) for each \( t \) does not guarantee that the recourses at different time points are compatible with each other. For example, there may be an injection solution for the 5-minute checkpoint and a solution for the 15-minute checkpoint, respectively, but it may not be feasible to ramp from the 5-minute solution to the 15-minute solution.

Appendix A provides a numerical example that concretely demonstrates this issue. To deal with this situation, we postulate an alternative feasibility condition for contingency \( k \) that couples the contiguous time points, as follows:

\[
\begin{align*}
g_k(x_k^t, u_k^t) &= 0 \\
h_k(x_k^t, u_k^t) &\leq \epsilon_t \\
|u_k^t - u_{k-1}^t| &\leq \Delta_t - \Delta_{t-1}
\end{align*}
\]

for \( t = 1, \ldots, T \) and \( u_k^0 = u_0 \). The resulting optimization problem is

\[
\begin{align*}
\min_{x, u} & \quad f_0(x_0, u_0) \\
\text{s.t.} & \quad g_0(x_0, u_0) = 0 \\
& \quad h_0(x_0, u_0) \leq 0 \\
& \quad g_k(x_k^t, u_k^t) = 0 \quad k = 1, \ldots, K, t = 0, \ldots, T \\
& \quad h_k(x_k^t, u_k^t) \leq \epsilon_t \quad k = 1, \ldots, K, t = 0, \ldots, T \\
& \quad |u_k^t - u_{k-1}^t| \leq \Delta_t \quad k = 1, \ldots, K, t = 1, \ldots, T \\
& \quad u_k^0 - u_0 = 0 \quad k = 1, \ldots, K
\end{align*}
\]

where \( \Delta_t = \Delta_t - \Delta_{t-1} \). This is a linear program when \( f \), \( g \) and \( h \) are all linear as defined above. In the remainder of the paper, we discuss computational techniques for efficient solutions of this LP. For illustration, we plot the sparsity pattern of a small problem instance in Figure 3. In the plot, the columns (variables) are arranged in the order \( \{u_0, x_0, u_1^0, u_1^1, x_1^T, u_2^0, x_2^0, \ldots\} \), and the rows (constraints) are arranged in the corresponding appropriate order. Note the inequalities \( h_k(\cdot) \leq \epsilon_t, \) i.e., (2) to (5), are handled as variable bounds hence do not appear in the matrix. It is apparent that the Jacobian is almost a band matrix if not for the constraints that link the control variable \( u_0 \) of the base-case with those of the contingency cases. Also note that the problem size grows linearly as the number of contingencies increases. These characteristics make the problem suitable for decomposition methods, which we discuss below.

III. Benders’ Decomposition

The common Benders’ decomposition scheme [14]–[16] reformulates model (8) into an equivalent form, as follows:

\[
\begin{align*}
\min_{x_0, u_0} & \quad f_0(x_0, u_0) \\
\text{s.t.} & \quad g_0(x_0, u_0) = 0 \\
& \quad h_0(x_0, u_0) \leq 0 \\
& \quad g_k(x_k^t, u_k^t) = 0 \quad k = 1, \ldots, K, t = 0, \ldots, T \\
& \quad h_k(x_k^t, u_k^t) \leq \epsilon_t \quad k = 1, \ldots, K, t = 0, \ldots, T \\
& \quad |u_k^t - u_{k-1}^t| \leq \Delta_t \quad k = 1, \ldots, K, t = 1, \ldots, T \\
& \quad u_k^0 - u_0 = 0 \quad k = 1, \ldots, K \\
& \quad s_k^t \geq 0 \quad t = 1, \ldots, T
\end{align*}
\]

where \( w_k(u_0) \) is the value function of the \( k \)-th sub-problem, given by

\[
w_k(u_0) = \min_{x_k, u_k, s_k} \|s_k^t\|
\]

\[
\text{s.t.} \quad g_k(x_k^t, u_k^t) = 0 \quad t = 0, \ldots, T \\
& \quad h_k(x_k^t, u_k^t) \leq \epsilon_t \quad t = 0, \ldots, T \\
& \quad |u_k^t - u_{k-1}^t| - s_k^t \leq \Delta_k \quad t = 1, \ldots, T \\
& \quad u_k^0 - u_0 = 0 \\
& \quad s_k^t \geq 0 \quad t = 1, \ldots, T
\]

Note that \( s_k \) is an artificial variable added to model the constraint violations. Since the sub-problem is a linear program, it follows from LP duality that any given point \( \tilde{u}_0 \) and the associated value \( w_k(\tilde{u}_0) \) (denoted by \( \tilde{w}_k \)) can provide a linear function of \( u_0 \) that underestimates \( w_k(u_0) \), as follows:

\[
w_k(u_0) \geq \tilde{u}_0 + \tilde{w}_k(u_0 - u_0) \geq 0
\]

where \( \tilde{w}_k + \tilde{\lambda}_k(u_0 - \tilde{u}_0) \leq 0 \) (15) is a necessary condition for (12) hence is a valid inequality for the master problem, and as a substitute for (12) which is hard to impose directly, it will cut off the point \( \tilde{u}_0 \) if \( \tilde{w}_k \) is positive.

The Benders’ decomposition algorithm alternates between solving the master problem and the sub-problems, hence approaches a better and better satisfaction of (12) until certain
convergence criteria are met. In each iteration, the master problem (9) - (11) with previously added cuts is solved. Subsequently the subproblems are solved one by one given the master solution. Each subproblem solution having a positive objective value will supply a new cut to the master problem for the next iteration. For a given problem instance, the number of variables in the master problem and size of the subproblem are fixed regardless of how many contingencies there are to consider. In this sense, the algorithm “decomposes” the big LP by approaching its solution via repeatedly solving smaller LPs.

It is worth noting that model (8) can be succinctly expressed as a two-stage stochastic program (SP) in the extended mathematical modeling (EMP) framework within the GAMS modeling software. An SP solver, e.g., DE and Lindo\(^1\), can then be called to solve the problem in both the deterministic equivalent form (big LP) and the Benders’ decomposition form. However, the Benders’ algorithm implemented in Lindo (which is the only general purpose Benders’ code available) is unable to handle any problem-specific structure or solve the subproblems in parallel. For example, it takes Lindo about 31 minutes to solve the 118-bus 183-contingency case using its Benders’ algorithm, even worse than the “Vanilla” Benders’ algorithm that we implemented directly in GAMS (computation times for this case is listed in the first row of Table II).

IV. COMPUTATIONAL ENHANCEMENTS

A. Formulation

The variable \( s^k_t \) in the subproblem captures the violation or infeasibility of the ramping constraints evaluated at the candidate solution \( u_0 \). In the literature [2], [3], [9], all authors used the \( L_1 \) norm in the subproblem objective function, i.e., minimizing the sum (over all buses) of violations. We find that the following modifications to the subproblem formulation reduce the number of Benders’ iterations required for convergence in many cases, as demonstrated in Figure 4. We will apply these modifications in all subsequent discussions.

- Use \( L_\infty \) norm of \( s^k_t \) in the objective, i.e., minimizing the maximum (over all constraints for which the violation variable is added) violation. This is aligned with the normalization idea of Fischetti et al. [17].
- Allow violation in the inequality constraints, i.e., substitute \( h_k(x^u_t, u^u_t) - s^k_t \leq 0 \) for \( h_k(x^u_t, u^u_t) \leq 0 \) in the subproblem. This also provides convenience in detecting infeasible contingencies, as will be discussed below.

We do not have a proof of the advantage of using \( L_\infty \) norm over using the \( L_1 \) norm. However, reductions in the number of iterations are consistently observed (although not always as significant as shown in the figure) and we have not encountered any case that takes more iterations using \( L_\infty \) rather than \( L_1 \). Since all known papers on SCED-C happened to explicitly adopt the \( L_1 \) norm in their formulations, we find it useful to report our observations here. As a side note, the 118-bus instance used in Figure 4 was made harder to solve (so that it takes Benders’ algorithm many iterations to converge) by using nodal loads 1.8 times higher than the original values.

B. Dealing with Infeasibility

In practical use of the SCED model (with or without the corrective rescheduling, single- or multi-stage), there is an implicit assumption (belief) that a feasible solution exists, i.e., the security constraints are satisfiable. Indeed, if it frequently occurs that no operating point is able to meet the security criteria, it probably indicates that the criteria are too restrictive and need a change. Realistically, not all lines in the network are included in the contingency list of SCED but only those having crucial importance, e.g., high-voltage backbone transmission lines, and those for which the consequence of failure is controllable (by which we mean “no load is lost”) via dispatch or rescheduling. Lines whose failure would island a load bus, for example, should not be included in the contingency list for SCED, because nothing can be done beforehand to avoid shedding load should the failure occur.

Despite the sensible selection of contingencies, there can be no a-priori guarantee for the existence of a feasible solution before actually running the SCED. Being notified that the model is infeasible is the last thing system operators want to see from a SCED run – in this case, they at least need to know what is causing the infeasibility, if not how to correct it. One way of avoiding infeasibility is to allow constraint violation (i.e., load shedding) and penalize it in the objective function, see, e.g., [18]. There are three drawbacks for this method: (1) Determination of the penalty factor is almost entirely arbitrary; (2) Shedding a load simply because its supply line MAY fail is impractical\(^2\); (3) Keeping a large number of penalty variables (i.e., one for each equality constraint) in the (master) model to prepare for infeasible situations which only occur occasionally is not efficient modeling practice.

Alternatively, we integrate in the Benders’ algorithm a mechanism to dynamically identify and remove the contingencies that would cause infeasibility. Denote the base-case load as \( u_0 \) and use the base-case feasible set by \( F = \{ u_0 \in \mathbb{R}^n | \exists x_0 \text{ such that } g_0(x_0, u_0) = 0 \} \).

\(^1\)The DE Solver is part of GAMS EMP/PSP framework and Lindo is an external solver to GAMS. Documentation can be found at http://www.gams.com/solvers/

\(^2\)This would happen if the only line that connects a load bus with the rest of the network were in the contingency list. A practical treatment should be to shed the load when the contingency actually occurs.
0, \ h_0(x_0, u_0) \leq 0\}, and the feasible set for contingency \( k \) by \( \mathcal{F}_k = \{u_0 \in \mathbb{R}^n | \exists (x_k^i, v_k^i) \text{ such that (7) holds}\}. \) Let us assume base case feasibility, i.e., \( \mathcal{F} \neq \emptyset \).

We call a contingency \( k \) intrinsically infeasible if \( \mathcal{F} \cap \mathcal{F}_k = \emptyset \). For such a contingency, the corresponding subproblem is either infeasible for all \( u_0 \in \mathcal{F} \), which indicates \( \mathcal{F}_k = \emptyset \), or optimal with a positive objective value for all \( u_0 \in \mathcal{F} \). In the former case, the subproblem will be infeasible in the first run (in the first iteration) and we can remove the contingency immediately\(^3\). In the latter case, the subproblem keeps generating cuts for the master problem until the master problem becomes infeasible due to conflicting cuts.

Another source of infeasibility comes from the case where multiple contingencies are not simultaneously feasible, e.g., \( \bigcap_{k=1}^{K} \mathcal{F}_k \cap \mathcal{F} = \emptyset \). Such a case manifests itself in the form of an infeasible master problem caused by conflicting cuts. The two cases of conflicting cuts are illustrated in Figure 5.

Our order of business is to remove the “problematic” contingencies whenever the master problem becomes infeasible. We do this by solving a modified master model, constructed by adding a nonnegative violation variable \( v_k^i \) to each of the previously added cuts as well as adding a linear term in the objective function to penalize the violation (with a penalty factor \( M \)). The solution of this model indicates the violated cuts (for which the violation variable is positive). We then remove any contingency that has contributed one or more violated cuts. The modified master problem is outlined below.

\[
\begin{align*}
\min_{x_0, u_0} & \quad f_0(x_0, u_0) + \sum_{(k,i) \in \text{CUT}} M v_k^i \\
\text{s.t.} & \quad g_0(x_0, u_0) = 0 \\
& \quad h_0(x_0, u_0) \leq 0 \\
& \quad \bar{w}_k^i + \bar{y}_k^i (u_0 - \bar{u}_0^i) - v_k^i \leq 0 \quad \forall (k,i) \in \text{CUT} \\
& \quad v_k^i \geq 0 \quad \forall (k,i) \in \text{CUT}
\end{align*}
\]

\(^3\) It must be the power balance constraint in the subproblem that makes it infeasible, since violation is allowed everywhere else. This corresponds to the situation where the contingency isolates a load node or sub-network from the rest of the network, a situation that is not considered insecure in the “N-1” security context.

where the superscript \( i \) indexes the iteration and \((k,i) \in \text{CUT}\) means that contingency \( k \) has generated a cut in iteration \( i \).

Note that by penalizing the “sparsity inducing” \( L_1 \) norm of the violation, we intend to approximately identify a minimal number of problematic contingencies whose removal would restore feasibility (which corresponds to a NP-hard problem). This approach is motivated by sparse optimization methods [19] and is shown to be effective in our experiments.

### C. Algorithmic Enhancement

The master problem is a small-sized (relative to the subproblems) linear program and is easy to solve. It is also easy to update the optimal solution from one iteration to the next, since adding cuts does not change the dual feasibility of an LP. Taking a 2383-bus case for example, it takes the CPLEX dual simplex method less than 1 second to solve the master model from scratch and takes less than 0.3 second for solution updates between successive iterations. In contrast, the majority of the solution time is spent on solving the many subproblems, each of which is also three times larger than the master problem (not counting the cuts), see Figure 3. We tailor several enhancement schemes to solving the subproblems using GAMS and CPLEX.

1) Reducing the number of LP runs: In practice, although a long list of contingencies need to be considered, most of them turn out not to be binding in the final SCED solution. In the Benders’ solution process, we observe via experiments that if a contingency is feasible (i.e., its subproblem has an optimal value of 0) in an iteration, it is also likely to be feasible in subsequent iterations. This observation can be exploited to improve the algorithmic design, as follows. At any iteration if a contingency becomes feasible, we temporarily exempt it from the feasibility test in subsequent iterations, which results in a shrinking list of active contingencies as the algorithm progresses. When this list is empty, we import the whole list of contingencies again and test the simultaneous feasibility of the current master solution. If it is feasible for all contingencies, the algorithm terminates; otherwise, the above process continues. This method could greatly reduce the total number of LP runs.

We apply this approach to the 118-bus case and demonstrate the solution process in Figure 6. There are 186 lines in the network and two of them (i.e., bus 12 to bus 117 and bus
68 to bus 116) are determined to be intrinsically infeasible by pre-screening, so we monitor the remaining 184 lines in the experiment. In iteration (iter) 1, 184 subproblems are “computed” among which 14 are “captured” to have a positive objective value. In iter 2, only those 14 subproblems captured in the previous iteration are computed, so in this fashion we are dealing with a shrinking list of active contingencies. In iter 6, two are computed but none is captured, which means the active list becomes empty. Therefore, in iter 7 the active list is reset to the whole list and three were captured. This process repeats until in iter 16 none is captured after computing for the whole list, which means that the current $u_0$ is feasible for all contingencies and the algorithm terminates. Note that in iter 3, contingency #146 (i.e., bus 85 to bus 86) is removed due to its causing infeasibility. The correctness of its removal has been verified by solving the full LP formulation.

2) Using barrier method without crossover for subproblems: CPLEX offers several options for LP methods, including primal simplex, dual simplex, network simplex, barrier, and concurrent. By default, CPLEX chooses the simplex method to solve the subproblem LPs. We found via experiments that the barrier method actually solves the subproblem faster than the simplex method for big instances (e.g., 2383-bus case). CPLEX barrier optimizer automatically invokes a crossover process when the barrier algorithm terminates, in order to produce a basic solution. However, any optimal solution, not necessarily a basic one, suffices to generate a valid cut (15). Therefore, we can turn off the crossover to save time, i.e., setting CPLEX option barcrossalg=1. Table 1 shows the potential time saving of this choice of solver options.

3) Solving batches of subproblems in parallel: Benders' decomposition algorithm is naturally amenable to parallel computing. We harness the multi-core hardware and multi-threading capability of the operating system to solve the subproblems in parallel during each iteration. Specifically, we evenly divide the active contingencies into $N$ batches, and run the batches on separate processors. In solving the series of subproblem LPs in each batch, we use the GUSS facility [20] within GAMS to shorten the overall solution time. As the subproblems are similar in structure, GUSS constructs the model rim once and plugs in different parameter data for different LPs. This eliminates the repetitive work of building the model from scratch for each LP, thus saves time.

4) Invoking feasibility checker and adding difficult contingencies to the master problem: Experiments on different realistic data sets provide the following observation. Between two successive whole-list scans (e.g., between iter 1 and 7 in Figure 6), most contingencies will be removed from the active list in a few (less than ten) iterations, but at times a small number of contingencies will remain in the active list for many iterations before becoming feasible or proven to be intrinsically infeasible (by the conflicting cuts of Figure 5(a)). Such contingencies incur extended computational costs in two ways: (1) Dealing with only a few subproblems per iteration is an inefficient use of parallel computing considering its overhead and (2) if the persistent contingency was intrinsically infeasible, all the iterations spent in detecting the infeasibility would be “wasted”. In other words, if infeasibility was detected earlier, much time could be saved. We mitigate these difficulties as follows. When the size of the active contingency list drops to a certain threshold level $L^k$, we initiate a “feasibility checker” job to run in parallel with the main Benders’ loop. This job checks on an individual basis the feasibility for all the contingencies on the active list, by solving for each contingency a reduced SCED model consisting of only the base case and the target contingency that is being checked. The feasibility checker (FC) model for contingency $k$ is as follows.

$$\begin{align*}
\min_{u_0, x_0, r^+, r^-} & \quad ||r^+ + r^-|| \\
\text{s.t.} & \quad g_0(x_0, u_0) + r^+ - r^- = 0 \\
& \quad h_0(x_0, u_0) \leq 0 \\
& \quad u_0 \in \mathcal{F}_k \\
& \quad r^+, r^- \geq 0
\end{align*}$$

The feasibility checker job is finished.

Table I

<table>
<thead>
<tr>
<th>Case</th>
<th>Subproblem Size</th>
<th>Diff</th>
<th>Split</th>
<th>Barrier</th>
<th>Barrier (\times)</th>
</tr>
</thead>
<tbody>
<tr>
<td>118-bus</td>
<td>1079</td>
<td>2608</td>
<td>8545</td>
<td>14.4</td>
<td>13.4</td>
</tr>
<tr>
<td>2383-bus</td>
<td>16814</td>
<td>37129</td>
<td>115006</td>
<td>453.6</td>
<td>139.0</td>
</tr>
</tbody>
</table>

V. Experiments

We use the IEEE 118-bus test case as well as several sets of the Polish network data for experiments. Although

---

The size of the active list may be smaller than $L^k$ by the time the feasibility checker job is finished.
these data sets are available from various sources, e.g., the Matpower package, the original data lack meaningful values for line thermal ratings and generator ramp rates, which are critical for the SCED with multi-stage rescheduling\(^5\). We adopt the data provided by a FERC project, which made up the missing values based on reasonable engineering assumptions and enriched the existing format [21]. In particular, the rateA, rateB and rateC of a branch are taken as the \(F^{\text{Normal}}\), \(F^{\text{LTE}}\) and \(F^{\text{STE}}\), respectively, associated with three post-contingency stages (checkpoints). The algorithms are implemented in the GAMS modeling software. Throughout the experiments, we use the feasibility tolerance of \(10^{-5}\), i.e., if the subproblem objective value \(||s_k||\) is lower than \(10^{-6}\), the contingency is considered feasible.

A. Comparison of Performance on Feasible Instances

Table II demonstrates the effect of different algorithmic enhancements on the solution speed. The “Big LP” formulation is solved by the simplex algorithm (CPLEX default) and the barrier method without crossover. Note that the crossover step may take significantly longer time than the core barrier algorithm. The subsequent columns of the table lists results (i.e., number of iterations taken, number of subproblem LPs solved and the total solution time in seconds) brought by incrementally added features (as described by the header of the column). The “Vanilla Benders” represents the original Benders’ algorithm with the formulation improvement described in Section IV-A. “RedLP+Opt” implements the first two enhancement schemes described in Section IV-C, i.e., implementing algorithmic control to reduce the number of LP runs and using appropriate solver options for subproblem speedup. “Paraguss”, in addition, solves the subproblems in (8 or 40, depending on problem size) parallel batches and uses the GUSS facility in each batch. “Fatmaster (5)” represent the complete set of features presented in this paper. In addition to “Paraguss”, it enables the feasibility checker (with \(L^c = 5\)) and adds difficult contingencies to the master model. All the implementations that solved the given case within the time limit (2 hours) obtained the same optimal objective value, listed in the last column.

Because neither the “Big LP” formulation nor the original Benders’ algorithm is capable of handling infeasible contingencies and an infeasible problem is not an interesting subject for comparison, we use a pre-screened list of lines as contingencies which are guaranteed to constitute a feasible SCED case. In Table II, the first five cases in the upper half of the table are small to medium-sized. For the 118-bus case, we monitor (meaning: set as contingency) all the 183 lines in the pre-screened list and for the 2383-bus case\(^6\), we monitor the first 20, 50, 100 and 400 lines, respectively, in the pre-screened list. These cases are run on a Dell laptop\(^7\). We use 8 parallel processes in the “Paraguss” implementation. It is apparent that each enhancement scheme yields substantial time savings.

The next five cases in Table II are the largest possible feasible instances on the corresponding network data (wp means winter peak and wop means winter off-peak, etc.), as we monitor the complete pre-screened list of lines. These cases are run on a Dell R710 server with two 3.46G X5690 Xeon Chips, 12 Cores and 288GB Memory. The “Big LP” formulation is unable to solve any of these cases within 2 hours due to the large model size. For example, the 2383-bus 2349-contingency case results in a 18GB LP for the solver. We ignored the uncompetitive “Vanilla Benders” algorithm on these big cases but instead ran the “RedLP+OPT” without the 2-hour time limit. It is worth noting that our final approach “Fatmaster” is able to solve all cases in 10 minutes and solve most cases well within 5 minutes.

B. Performance on Possibly Infeasible Instances

Given a network case and a list of contingencies, it is not known a priori whether the data represents a feasible SCED instance or not. A straightforward first step to “purify” the data is pre-screening the intrinsically infeasible contingencies one by one, which involves solving the model (FC) for each contingency \(k\) in the list. Table III lists the results of this process run on all lines in the network. We can see that pre-screening is very time-consuming. Even if parallel computing (e.g., 100 processors) were utilized, it would still take several hundred seconds to pre-screen a large network case.

In contrast, our approach of dealing with infeasibility takes little extra time and is able to identify infeasible cases (as in Figure 5(b)) to which the pre-screening is blind. This is demonstrated in Table IV. The upper half of the table are experiments that include all lines in the initial contingency list, i.e., the “N-1” cases. The column “Added” lists the number of contingencies that have been added to the master problem in the solution process. The numbers are small, indicating

\(^5\)The original data may have all otherwise non-trivial contingencies feasible for any base-case dispatch, which would make experiments unilluminating.

\(^6\)In the Polish network case names, the suffix “wp” means winter peak, “sop” means summer off-peak and so on.

\(^7\)Dell precision M4500 with Intel Core i7 CPU Q840 @1.87GHz, 8GB RAM, on Windows 7.
that the choice of $L^G = 5$ is appropriate for these cases. The column “Tabu” lists the number of of contingencies that have been removed during the run due to infeasibility. The lower half of the table are experiments that only takes the pre-screened lines (the “Feasible” lines coming from Table III) as contingencies. We can see that the solution times of the two cases do not differ much, which means that our approach of removing infeasibility is much more efficient than pre-screening. Furthermore, the fact that 4 extra lines (other than those identified by the pre-screening) are removed in the 2383-bus case indicates that (1) Pre-screening is indeed unreliable in practice and (2) our method is effective at approximating a minimal set of problematic contingencies when the problem is infeasible.

Table V provides the list of binding contingencies at the optimal solution of the “N-1” cases. A contingency is identified as binding at the solution if any cut it contributes has a nonzero multiplier value, or the ramping constraint (13) has a nonzero multiplier if the contingency has been added as a master model. In the table, numbers in bold face are binding contingencies in the master model. We can see that very few contingencies are binding at the optimal solution, although a larger number of contingencies have been active along the algorithm iterations. Note that Table V only lists the contingency numbers. Interested readers may find a mapping of these contingency numbers to specific lines in Appendix A.

**VI. CONCLUSION**

Incorporating the post-contingency rescheduling actions into the SCED model provides better economic efficiency but also increases the computational difficulty which hinders its industrial application. Our work contributes to the advancement of the SCED-C research in both modeling and computation aspects. First, we have proposed a model to correctly address the multiple stages of rescheduling requirement found in realistic operating procedures. Second, we have devised a series of computational enhancements to solve the proposed model. The enhanced algorithm, coded directly in GAMS and available for general use without need of specialized hardware, was shown to be much faster than the original Benders’ algorithm and was able to solve large instances within reasonable amount of time. Finally, our computational results could serve as an estimate on how far/close the current technology is to the industrial deployment of the multi-stage SCED-C model. Future work will quantify the economic benefit of multi-stage rescheduling and investigate the application of this solution approach to an online setting.

**APPENDIX A**

**EXAMPLE OF INCONSISTENT RECOURSES**

The example is given in Figure 7. In the pre-contingency state, the 150 MW load at Bus 1 is supplied by Line 1 and Line 2. After the contingency (outage of Line 1) occurs, STE and LTE ratings of Line 2 requires the generators (G1, G2 and G3) to provide a total ramp-up of 50 MW in 5 minutes and 75 MW in 15 minutes, respectively. The STE rating requirement can be satisfied via ramping up G1 by 10 MW and ramping up G2 by 40 MW (of which 20 MW will flow on L3 and another 20 WM will flow on L4 and L5). At this stage, the flow on L3 has reached its capacity of 20 MW, so we cannot continue to ramp up G3 to meet the LTE requirement. Note that the parameter $x$ on a line indicates the line’s reactance, which dictates how the power flow distributes along different paths.
### APPENDIX B

**ACTIVE CONTINGENCY MAPPING AT OPTIMUM**

As a companion of Table V, Table VI provides the mapping from a contingency number \( k \) to line identification \((i, j, c)\), in the form of \( k : (i, j, c) \). The circuit number \( c \) is omitted with a note that all the lines listed here has a circuit number \( c = 1 \). For example, the first entry “43: (26,30)” in the 118-bus case reads “contingency #43 is the outage of the line (circuit #1, if there are multiple) connecting bus 26 and bus 30.”

<table>
<thead>
<tr>
<th>Contingency Mapping at Optimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>118-bus</td>
</tr>
<tr>
<td>118-bus</td>
</tr>
<tr>
<td>118-bus</td>
</tr>
<tr>
<td>118-bus</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>118-bus</th>
<th>2383-bus</th>
<th>2736-bus</th>
<th>2746-bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>43: (26,30)</td>
<td>344: (310,6)</td>
<td>170: (131,75)</td>
<td>3: (7)</td>
</tr>
<tr>
<td>63: (38,65)</td>
<td>414: (367,14)</td>
<td>292: (131,75)</td>
<td>4: (8,14)</td>
</tr>
<tr>
<td>124: (71,73)</td>
<td>546: (477,420)</td>
<td>576: (508,361)</td>
<td>440: (395,21)</td>
</tr>
<tr>
<td>184: (110,111)</td>
<td>1798: (1514,894)</td>
<td>573: (503,493)</td>
<td></td>
</tr>
<tr>
<td>185: (110,112)</td>
<td>2164: (1845,135)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### ACKNOWLEDGMENT

This work is partially supported through a contract from Argonne, a U.S. Department of Energy Office of Science Laboratory, and was supported by the Applied Mathematics activity, Advanced Scientific Computing Research program within the DOE Office of Science, along with contracts from the DOE Office of Electricity Delivery and Energy Reliability and the Federal Energy Research Commission.

### REFERENCES


Yanchao Liu (S’13) received B.S. degree in industrial engineering (IE) at Huazhong University of Science and Technology in 2006, and M.S. degree in IE at the University of Arkansas in 2008. He is a Ph.D. student of operations research at the University of Wisconsin-Madison. His research interest includes optimization modeling of power systems and energy markets.

Michael C. Ferris is Professor of Computer Sciences and leads the Optimization Group within the Wisconsin Institutes for Discovery at the University of Wisconsin-Madison. He received his Ph.D. from the University of Cambridge. Dr. Ferris’ research is concerned with algorithmic and interface development for large scale problems in complementarity and mathematical optimization, including links to the GAMS and AMPL modeling languages, and general purpose software such as PATH, NLPEC and EMP. He is a fellow of INFORMS and SIAM.

Feng Zhao (M’08) received B.S. in Automatic Control from Shanghai JiaoTong University, China in 1998, M.S. in Control Theory & Control Engineering from Tsinghua University, China in 2001, and Ph.D. in Electrical Engineering from University of Connecticut, USA in 2008. Currently he is a Lead Analyst/Engineer at ISO New England. His research interests include mathematical optimization, power system planning & operations, and economics of electricity markets.