Adaptive Online Control of Cascading Blackouts

Daniel Bienstock, *Member, IEEE*

***Abstract*—We study online control algorithms to be deployed in the event of a cascading power system failure. The control mechanism is computed after the initial event that sets off the cascade and is applied as the cascade unfolds, with the goal of reaching a stable state while shedding a minimum amount of load. We focus on robust controls, using models of line outages that explicitly account for noise. Computational experience is presented using simulated cascading failures of the U.S. Eastern Interconnect.**

***Index Terms*—Mathematical Programming, Network Theory, Optimization, Power system analysis computing, Power generation dispatch, Power system faults, Power System Security, Robust control, Smart grids.**

I. Introduction

CASCADING failures of large-scale power grids are rare events that nevertheless pose a grave and likely growing risk to society and national security. The August 2003 blackout in the northeast U.S. and Canada, and other events worldwide stress that while such events are infrequent their cost can be significant. It is estimated that more cascading blackouts can be expected, despite the sophistication and robustness of modern grids.

Cascading failures are typically caused by subtle and difficult to predict *combinations* of individual events such as single line outages. As a result, a significant amount of research has tackled the problem of identifying compound contingencies (such as the outage of a few lines) that could place a grid in an unstable condition. This is known as the $N-k$ problem (where the small integer k represents the number of simultaneous faults being considered). It is intrinsically a combinatorial problem, which, given the large size of national grids, is quite challenging. See [2], [6], [7], [19] and references therein. $N-k$ studies can help discover and correct severe contingencies, but of course there is no guarantee that *all* vulnerabilities will be successfully overcome. An $(N-3)$-secure system may be (probably will be) very robust, even beyond the 3-fold guarantee, but vulnerabilities may remain, for example if they are of a type outside of the scope of faults being considered. Thus, effective control in the event of a cascade remains an important goal.

In this paper we describe efficient algorithms that, should a potentially significant contingency be realized, can efficiently compute adaptive controls with the goal of averting a catastrophic cascade. We assume a cascade that at the onset is slow moving; therefore a nontrivial amount of time is initially available for significant computing (as arguably was the case in the 2003 cascade). The fact that we compute the control *after* the initiating event for the cascade implies an exponential reduction in combinatorial complexity; however our models simulate the evolution of the grid as the cascade progresses and computational challenges remain.

Our controls take the form of adaptive load shedding and/or generator redispatch. In the load shedding case, the control laws specify, as a function of time and as a function of current state data (such as line overloads), how much load to shed at each load bus. Thus, the controls are both adaptive and distributed. An additional goal in the computation of the control is that it should be *robust* so as to accommodate the expected large amount of noise that would be present during a cascade; in particular when modeling line outages. The starting point for our work are models for cascades due to Carreras, Lynch, Newman and Dobson (see [9], [10], [11], [12]). The models are modified by adding a computation step (that takes place immediately after the initiating event for the cascade) where the control is computed and by inserting a control step at each stage of the cascade.

We present computational experiments with simulated cascades on the U.S. Eastern Interconnect using a parallel implementation of our control computation algorithm. The experiments show that very significant percentages of demand can be maintained even in the face of severe cascades.

II. Models

The experiments we describe in this abstract use the *linearized* approximation to the power flow problem ([3], [4]). This is only done so as to make it possible to carry out very high volume of large-scale power flow computations fast and reliably, even under conditions where a grid is somewhat compromised. By the time of the conference we plan to report on experiments using more comprehensive AC power flow models. In the *linearized* approximation to the power flow problem, we are given a directed graph G with n nodes and m arcs (corresponding, respectively, to buses and lines). In addition

- For each arc j we are given two positive quantities: its *flow limit* u_j and its *reactance* x_j.
- We are given a *supply-demand* vector $\beta \in \mathcal{R}^n$ with the following interpretation. For a node i, if $\beta_i > 0$ then i is a *generator* (a source node) while if $\beta_i < 0$ then i is a *load* (a demand node) and in that case $-\beta_i$ is the *demand* at i. The condition $\sum_i \beta_i = 0$ is assumed to hold. For a generator node i, we indicate by the constant $\tilde{s}_i$ the maximum supply of i. We denote by $\mathcal{G}$ denote the set of generators and by $\mathcal{D}$ the set of demand nodes.

The linearized power flow problem specifies a variable f_{ij} associated with each arc j (active power flow) and a variable ϕ_i

D. Bienstock is with the Departments of Industrial Engineering and Operations Research, and Applied Physics and Applied Mathematics, Columbia University, New York, NY, 10027 USA e-mail: (see http://www.columbia.edu/∼dano).

Partially funded by DOE award DE-SC000267

associated with each bus i (phase angle). The problem consists in finding a solution to the system of equations:

$$Nf \;=\; \beta, \quad N^T\phi - Xf \;=\; 0, \tag{1}$$

where N denotes the node-arc incidence matrix of G [1] and $X = \mathbf{diag}\{x_{ij}\}$.

We model cascades using Template 2.1, which draws from [9], [10], [11]. The template assumes that an event that sets-off the cascade has taken place. Further, in this paper we focus on line outages, however the template and our algorithms below are easily modified to accommodate wider types of faults.

```
Template 2.1: GENERIC CASCADE TEMPLATE
Input: a power grid with graph $G$ (post-initiating event).
Set $G^1 = G$.
For $r = 1, 2, \ldots$ do
    (comment: round $r$ of the cascade)
    1. Set $f^r$ = vector of power flows in $G^r$.
    2. Set $\mathcal{O}^r$ = set of lines of $G^r$ that become outaged
       in round $r$.
    3. Set $G^{r+1} = G^r - \mathcal{O}^r$. Adjust loads and
       generation in $G^r$.
```

The discretization of time in the template implicitly represents time scale which is appropriate for the application of control. We will discuss this point at greater length below. G^r is the grid at the start of round r and f^r indicates the power flow vector at the start of round r.

In order to make the template precise we need to provide a mechanism for the outage of lines (Step 2) and for the adjustment in Step 3, especially in the event of "islanding".

In terms of Step 2, let $0 \le \alpha \le 1$ be a fixed parameter, and for each line j and each round r define $\tilde{f}^r_j$ by

$$\tilde{f}^r_j \;=\; \alpha_j |f^r_j| + (1 - \alpha_j)\tilde{f}^{r-1}_j, \tag{2}$$

with $\tilde{f}^0_j$ set to the absolute value of the flow on j prior to the incident that initiates the cascade. The quantities $\tilde{f}^r_j$ are moving averages of the (absolute value) of the flow on line j; the parameter α_j serves to encode "memory" (for example, so as to model thermal effects). Using a value of α_j close to 1 yields a system with little memory or, equivalently, one where each round represents a "long" period of time, and conversely α_j close to 0 yields more memory, corresponding to shorter rounds.

A **simple** deterministic outage rule is to declare that

$$\text{line } j \text{ becomes outaged if } \tilde{f}^r_j > u_j. \tag{3}$$

From a computational perspective, such a rule can prove numerically unstable and can lead to non-robust models. It is worth stressing this point from two different perspectives: first, in applying (3) we compare two numerical quantities, one of which was obtained by a floating-point algorithm. Thus, the potential for round-off is certain. This problem can be somewhat be lessened by either "padding" (increasing) or "tightening" (reducing) the parameter u_j. In any case, the second problem with (3) is more severe: in cases where $\tilde{f}^r_j$ (or f^r_j) is near u_j, the outcome (outage or not) should be treated as ambiguous, because our cascade model does not incorporate all possible sources of "noise" or poorly understood phenomena, which in the course of the cascade could effectively reduce u_j (or increase the flow). That is to say, a control algorithm based on direct application of (3) would not be robust.

To overcome these difficulties, we introduce a stochastic modification of the deterministic rule, as follows.

Rule 2.2: STOCHASTIC LINE OUTAGE
Parameters: $0 \le \epsilon_r \le 1$ for each round r.
Notation: refer to Template 2.1 and equation (2).
Application: For a line j in G^r:

$$\text{if} \quad u_j < \tilde{f}^r_j, \;\; \text{then} \;\; j \in \mathcal{O}^r, \tag{4}$$

$$\text{if} \quad (1-\epsilon_r)u_j < \tilde{f}^r_j \le u_j, \quad \text{then} \;\; j \in \mathcal{O}^r \;\; \text{with probability } \tfrac{1}{2}, \tag{5}$$

$$\text{if} \quad \tilde{f}^r_j \le (1-\epsilon_r)u_j, \quad \text{then} \quad j \notin \mathcal{O}^r. \tag{6}$$

The random choice in (5) is an indirect way to incorporate some of the (poorly defined) "noise" mentioned above; additionally, from a mathematical perspective, it serves to smooth the cascade process. Typically we would have $\epsilon_1 \le \epsilon_2 \le \ldots$, indicating increasing uncertainty as the cascade progresses. If $\epsilon_r = 0$ for all r we obtain the pure deterministic rule.

The adjustment in Step 3 is necessary in the case of *islanding* i.e. the creation of several connected components. Any imbalance between supply and demand in an island of G^{r+1} must be corrected. Several mechanisms for effecting such balancing are plausible; for the sake of conciseness in this paper we assume that if in an island capacity generation exceeds demand, then the output of all generators in that island will be proportionally decreased so as to match demand (and viceversa) .

III. Adaptive Control

```
Template 3.1: CASCADE CONTROL
Input: a power grid with graph $G$. Set $G^1 = G$.

Step 0. Compute control algorithm.

For $r = 1, 2, \ldots, R-1$, do
    (comment: controlled round $r$ of the cascade)
    1. Set $f^r$ = vector of power flows in $G^r$.
    2. Observe state of grid (from state estimation).
    3. Apply control.
    4. Set $g^r$ = vector of resulting power flows in $G^r$.
    5. Set $\mathcal{O}^r$ = set of lines of $G^r$ that become outaged
       in round $r$.
    6. Set $G^{r+1} = G^r - \mathcal{O}^r$. Adjust loads and generation
       in $G^r$.

Termination (round $R$). If any island of $G^R$ has line
overloads, proportionally shed demand in that island
until all line overloads are eliminated.[a]
```

[a]The criterion of "stability" inherent in the termination step may obviously be incomplete when using a more complete model of power flows than the linearized model.

There are many ways to state an optimization problem that operates in the framework of Template 2.1 so as to compute some optimal behavior in the face of a cascade. Our focus is on adaptive controls that can be computed in real-time (at the start of the cascade) and applied during the cascade.

We modify Template 2.1 so as to incorporate the computation and application of control. Additionally, we will force the cascade to terminate in a fixed number R of rounds. This is necessary for computational purposes; additionally this rule embodies the (arguably desirable) outcome of controlling the cascade "quickly" with an agnostic termination criterion. When using the linearized power flow model, at round R we terminate the cascade by shedding demand so as to eliminate all line overloads (if necessary).

In Template 3.1 f^r indicates the power flow vector at the start of round r, and g^r is the power flow vector after application of control in round r. To make the template precise we will next describe an explicit control mechanism. The control will first be stated in its general form; later we will discuss special versions.

Our control takes the form of an affine law, described by a triple of values (c^r_v, b^r_v, s^r_v) (computed in Step 0 of Template 3.1) for each round r and load bus v. At round r, let d^r_v denote the current demand at v, and let κ^r_v be the maximum line overload in the island currently containing v. Then, in Step 3, bus v resets its demand to:

$$\min\{1,\, [b^r_v + s^r_v(c^r_v - \kappa^r_v)]^+\}\; d^r_v, \qquad \text{if } \kappa^r_v > c^r_v \tag{7}$$

$$d^r_v, \qquad \text{otherwise.} \tag{8}$$

(In equation (7), $[x]^+$ denotes $\max\{x, 0\}$.) We can provide some intuition for this rule. First, the scaling rule (7) can be roughly approximated by

$$\text{new demand} \;=\; \left(\, b^r_v + s^r_v(c^r_v - \kappa^r_v) \,\right) d^r_v. \tag{9}$$

(the purpose of the "min" and the "$^+$" in (7) is to ensure that we do not increase demand or set it to a negative value). (9) is an affine control law that sheds load in proportion to κ^r_v. Note that if we choose $b^r_v = c^r_v = 1$, and $s^r_v = 0$, then no control is applied. Other candidates for κ^r_v (other than maximum line overload) are plausible, such as *average* line overload, or a measure of mismatch between current power flows and (historically) stable values, however the maximum overload version is the only one we report on in this paper. In forthcoming work involving AC power flows, we plan to report on versions of κ^r_v that account for voltage "sag" and phase angle drift.

The *goal* of the control is to maximize the amount of demand being delivered by the grid after the termination step (round R). This is a nonsmooth, multistage optimization problem (see Section V). Additionally, national or regional grids can be quite large. Below we will discuss methodologies we use to address this problem in a computationally practicable way. First we will present experimental results using special cases of the control.

IV. First set of experiments

For our experiments we used a snapshot of the U.S. Eastern Interconnect, with approximately 15000 buses, 23000 lines, 2000 generators and 6000 load buses. The snapshot includes generator output levels, demands, (most) line flow limits, and other physical parameters for the lines.

In all the experiments the same approach was employed: first, we interdicted the grid according to a synthetic contingency, then we computed our affine control, and finally we studied the behavior of this control.

To obtain contingencies we used the following methodology, which removes a a set of K random, high power flow lines from the grid, while preserving connectivity. Here K is a given, small integer, and as before m is the number of lines.

(1) A spanning tree T is computed.
(2) Let $\hat{f}$ denote the power flow vector corresponding to the given demands and generator outputs. Renumber so that $|\hat{f}_1| \geq |\hat{f}_1| \ldots \geq |\hat{f}_m|$.
(3) Let $0 < \pi < 1$. Run steps (a) and (b), initialized with $S = \emptyset$, until stopping in (b):
    For $j = 1, \ldots, m$,
    (a) If line $j \notin T$, then
        with probability π reset $S \leftarrow S \cup j$.
    (b) If $|S| = K$, stop.
(4) The set S of lines is removed from the network, producing network G in our cascade template.

We used values of K ranging from 1 to 50, and for π we used values ranging from .1 to .5.

The purpose of the experiments that we report on below is to show the structure of a simple (though suboptimal) control of the form (7)-(8), and how the application of control affects the evolution of the cascade, as compared to the no-control situation.

We considered a case with $K = 2$ (two lines removed) and $R = 20$ rounds. In the computation of the moving averages of line overloads (eq. (2)) we used $\alpha = 0.9$.

First we consider the pure deterministic case of line outages, that is to say we use line outage rule (2.2) with $\epsilon_r = 0$ for all r. If no control is applied, then at the end of round of round 20 the *yield* (percentage of demand still being served) is 2.47%.The cascade is characterized by extremely high line overloads; see Table I (we will discuss implications of this below). At the start of round 1, in fact, the maximum line overload is 40.96, indicating that, likely, several lines with low flow limits are overloaded.

We searched for the best control of the form (7)-(8) where
(i) $c^r_v = b^r_v = 1$ for all v and r.
(ii) $s^r_v = 0$ for all v and $10 < r$. Thus, no control is applied after round 10.
(iii) For each $1 \leq r \leq 10$, either $s^r_v = 0.005$ for all v, or $s^r_v = 0$ for all v.

Thus we simply want to decide *when* to apply a control of a very simple form. Further, we are restricted to applying control in the first half of the cascade; this is done as protection against uncertainty in the later rounds of the cascade The rationale for the numerical values in (iii) is that $1 + 0.005 * (1 - 40) \approx 0.80$, that is to say, the application of this control in round 1 will "only" shed 20% of the demand.

We want to stress that the experiments in this section do not amount to a rigorous attempt at optimizing control. In fact, the control obtained through (i)-(iii) is only near-optimal. Instead we are trying to provide an example of the difference between an adequate control and the no-control option, and the questions that arise from the comparison. In particular, the amount 0.005 was arrived at through a simple grid-search process. See Section V for optimization methodologies.

In any case, the optimal control that satisfies conditions (i)-(iii) (and which we shall refer to as **c20** for future reference) attains a termination yield of 75.2%, picks rounds 2 and 7 to apply control. Note that since the maximum line overload is high in round 1, c20 allows some lines to become outaged in round 1.

This point is further elaborated in Table 1, where "r" indicates round and for each round, "κ" indicates maximum line overload at the start of the round, "O" is the number of lines outaged during the round, "I" is the number of islands at the end of the round and "Y" is the (rounded) percentage of demand being delivered end of the round. We stress that we count *all* islands, even those that consist of a single bus with no demand, and when computing the maximum line overload we consider *all* lines, no matter how minor.

TABLE I
Cascade evolutions

	No control				c20			
r	κ	**O**	**I**	**Y**	κ	**O**	**I**	**Y**
1	40.96	86	1	100	40.96	86	1	100
2	8.60	187	8	99	8.60	165	8	96
3	55.51	365	20	98	61.74	303	17	96
4	67.14	481	70	95	66.63	408	44	94
5	94.61	692	149	93	131.08	492	94	93
6	115.53	403	220	91	112.58	416	146	90
7	66.12	336	333	89	99.62	326	191	78
8	47.83	247	414	87	60.95	227	248	77
9	7.16	160	457	85	32.50	72	279	76
10	7.06	245	542	84	9.50	43	292	76
11	37.55	195	606	83	45.28	35	303	76
12	13.04	98	646	82	11.60	10	306	76
13	22.61	128	688	82	3.88	6	310	75
14	10.64	107	715	81	1.46	4	312	75
15	5.03	64	721	81	1.34	1	312	75
16	84.67	72	743	80	1.13	1	312	75
17	32.15	52	756	80	1.38	2	312	75
18	6.50	43	763	80	1.26	1	312	75
19	9.97	85	812	80	0.99	0	312	75
20	32.34	39	812	2	0.99	0	312	75

Discussion. We see that initially both cascades have very high line overloads, many line outages and large amounts of islanding. However, under c20 after line 11 the overages are significantly smaller, and rapidly decreasing, and after round 8 the number of new outages and islands is also much smaller (and decreasing); both in spite of the fact that control is last applied in round 7. Thus, effectively, the cascade has been "stabilized" under c20, long before the end of the time horizon.

The reader might wonder about the rapid decrease of yield from 80% to 2% in the no-control case. This is due to the termination feature in our cascades that requires all line overloads to be eliminated by the end of the last round; since the no-control cascade has very high maximum overload (32.34), at the start of round 20, the termination rule forces a drastic reduction in yield.

Nevertheless, in the no-control case, the combination of comparatively high yield (up to round 4), high number of line outages, large line overloads and large amount of islanding suggest the possibility that many of the outages involve unimportant lines, and likewise with many of the islands (though of course a 20% yield loss should indicate a severe contingency). One wonders if somehow the no-control option *might* be attractive if *enough* time (i.e., rounds) were available.

TABLE II
Further evolution of no-control cascade from Table I

r	**25**	**28**	**29**	**30**	**31**	**32**	**33**	**34**
O	21.63	2.00	5.70	2.50	2.38	1.35	1.07	0.99
Y	79	78	78	78	78	78	78	78

To investigate these possibilities, we extended the no-control cascade. Table II shows the results for selected rounds. We see that the no-control approach finally yields stability by round 34, attaining yield 78%. This is slightly better (but very close) to what c20 obtained in 20 rounds (and, furthermore, control action under c20 was restricted to rounds 1-10). Nevertheless, the no-control approach experiences significant line overloads as late as round 32.

By maintaining high overloads into very late rounds, the no-control strategy becomes more exposed to the unavoidable *uncertainty* that should be taken into account when modeling cascades, and which we have up to now ignored. We model noise by means of fault outage rule (2.2). In the following set of tests we assume that

$$\epsilon_r \quad = \quad 0.01 + 0.05 * \lfloor r/10 \rfloor. \qquad (10)$$

Possibly, noise should be increasing at a faster rate than the above formula stipulates (perhaps exponentially). However, the control considered in Table I as well as the no-control approach are both exposed to significant amounts of noise after round 10; more so in the no-control case. We would thus expect that under rule (10) the no-control approach will perform much more poorly.

To test these hypothesis, we ran 1000 simulations of cascades under rule (10) for the no-control case and for control using c20. The results are summarized as follows: using c20, the average yield is 42.90 and the standard deviation of yield is 27.47, whereas using no control the average yield is 7.96 and the standard deviation is 9.33. In other words, c20 proves much more robust than the no-control strategy, which is not surprising given the structure of rule (10). A question that arises as a result is whether c20 is in some sense optimally robust.

One way to investigate this question is to investigate controls that are less exposed to uncertainty by restricting them to a shorter timeline, i.e. by enforcing termination before round 20. For $T = 10, 15, 25$, we compute an optimal control required to terminate by round T, and otherwise subject to rules (i)-(iii), that is $c_v^r = b_v^r = 1$ for all v and r, $s_v^r = 0$ for all v and $10 < r$, and for each $1 \leq r \leq 10$, either $s_v^r = 0.005$ for all v, or $s_v^r = 0$ for all v. We name these controls c10, c15 and c25, respectively.

Table III presents the comparisons between all the options we have considered. In this table, "DetY" is the yield in the deterministic case ($\epsilon_r = 0$ for all r), "MaxY" and "MinY" are the maximum and minimum yields in all the simulations (resp.), "AveY" is the average yield and "StddY" is the standard deviation of yield.

TABLE III
Robustness comparison - 1000 runs using stochastic outage rule (2.2) with noise as in (10)

Option	DetY	MaxY	MinY	AveY	StddY
c10	37.49	39.05	0.00	11.81	11.84
c15	72.44	71.85	0.00	33.94	22.51
c20	75.19	76.30	1.17	41.90	27.47
c25	77.23	42.34	1.38	11.99	10.97
no control	77.75	36.04	0.00	7.96	9.33

Control c20 emerges as superior over c15 and c10. This can be explained as follows. Even though c15 and c10 are significantly less exposed to risk than c20, they are also restricted to operating, and terminating, during a stage of the cascade characterized by extremely high line overloads. Control c20, by being able to operate over 20 rounds, has "more time" while also avoiding the large uncertainty rounds 20 and higher. For this reason, c20 is also superior to c25 (their averages are separated by more than one standard deviation). One common feature that emerges in controls c10, c15, c20 and c25 (not shown in the table) is that no control is taken in round 1, and control is taken in round 2 (and in the cases of c10, c15 and c20, rounds 5 or 7).

We stress that (10) is *one* categorization of noise. Using a different formula the outcome could be different, say c15 could prove best. However, the outlook we are taking here is that by computing *a* robust control with respect to *some* rule such as (10) we obtain a control that remains robust (though possibly not optimally so) even if the model for uncertainty were to be somewhat changed. And, in any case, computing a control which is is somewhat robust should be better than completely ignoring uncertainty.

To explore these issues, we study the following model

$$\epsilon_r \quad = \quad 0.01 + 0.005 * r, \tag{11}$$

which can be considered a smoothed version of (10). Under this model both c15 and c20 are exposed to more noise than c10, and more noise than under rule (10). Consider Table IV. We see that c20 still appears superior to the other controls,

TABLE IV
Robustness comparison - 1000 runs using stochastic outage rule (2.2) with noise as in (11)

Option	DetY	MaxY	MinY	AveY	StddY
c10	37.49	38.93	0.00	7.54	9.55
c15	72.44	63.94	3.41	28.02	17.94
c20	75.19	73.04	0.00	32.24	21.30
c25	77.23	54.62	0.25	16.84	12.66
no control	77.75	18.86	0.00	5.11	5.28

though c15 is almost as good.

The above experiments do not amount to a full optimal robust control computation. Methodologies for computing robust controls are discussed in the next section.

V. Optimization methods

Given a control vector (c, b, s), denote by $\tilde{\Theta}^R(c, b, s)$ the final demand at termination of the R-round cascade controlled by (c, b, s). Our goal is to maximize $\tilde{\Theta}^R(c, b, s)$ over all controls. This is a nonconcave, in fact very combinatorial, maximization problem [8], [18]; it is very large (e.g. if $R = 10$ the (c, b, s) vector has more than 180000 variables in the case of the Eastern Interconnect). It is also important to incorporate stochastics.

In principle, the deterministic case of our problem could be tackled using mixed-integer programming techniques, and the stochastic version, using stochastic programming [16]. Of course, one could choose a different formulation of the cascade control problem than the one we chose (using a different kind of control, for example). But any formulation will have to deal with the combination of combinatorics in the network dynamics, multistage behavior, stochastics and very large size. In our opinion, this combination places the problem outside the capabilities of current optimization methodology, even in the deterministic case. We remind the reader that we envision our control as being computed in real time and we might only have one hour, or less, to do so.

Another point to stress is that nonconcavity in a maximization problem leads to non-monotone behavior: in our case, just because a small change in control leads to an improvement does not imply that a larger change will result in greater improvement.

In our approach, a basic method we rely on is outlined in Procedure 5.1.

```
Procedure 5.1: FIRST-ORDER ALGORITHM
Input: a control vector $(c, b, s)$.
For $k = 1, 2, \ldots$ do
    1. Estimate $g = \nabla\tilde{\Theta}^R(c, b, s)$.
    2. Choose "step-size" $\mu \geq 0$ and update control to
       $(c, b, s) + \mu * (g_c, g_b, g_s)$.
    3. If $\mu$ is small enough, stop.
```

This is a common first-order (steepest-ascent) method. In the deterministic case, Step 1 should be interpreted as a an approximate rule since $\tilde{\Theta}^R$ is not differentiable (our stochastic outage rule 2.2 does smooth out the expectation). The vices of procedure 5.1 are well known: even if $\tilde{\Theta}^R$ were smooth, its nonconcavity implies that the steepest-ascent method may not converge to a global optimum. And even if $\tilde{\Theta}^R$ were smooth and concave, steepest ascent may zigzag or stall. See [18].

In summary, Procedure 5.1 should be viewed as a *local search* method with which to explore the neighborhood of a solution. Finally, in our setting the procedure could prove expensive, since each evaluation of $\tilde{\Theta}^R$ (including in the estimation of $\nabla\tilde{\Theta}^R$ through finite differences) requires a cascade simulation, each round of which requires two power flow computations in our setup.

On the positive side, however, the procedure is flexible enough to handle (at increased computational cost) important features, such as more realistic AC power flow models, or more complete renditions of low-level controls in the operation of a power grid. Essentially, Procedure 5.1 is an example of simulation-based optimization, i.e. it only needs to have a "black-box" that computes the function $\tilde{\Theta}^R$.

An active research field that considers optimization under such assumptions is that of *derivative-free optimization* (see [13]) and related methods that incorporate second-order information [22]. In our estimation, these methodologies may not scale well to problems of the size we consider. In forthcoming work we will experiment on adaptations of these methodologies to our problem.

When we consider a model that includes stochastics, the first-order method can be viewed as a *stochastic gradients* algorithm (see [20], [17] – an alternative methodology is provided by bundle methods). In the stochastic gradients approach, a fixed *sample path* of the appropriate random variables is chosen in advance of each gradient and step-length computation. Care must be taken to ensure that this idea is properly implemented. In our implementation, we simulate one cascade Ω under the current control vector (c, b, s) and then compute the change in yield resulting from infinitesimally small deviations away from (c, b, s) on cascades with *precisely* the same sequence of line outages as Ω. It is outside of the scope of this writeup to describe this idea in complete detail.

Whether we use the stochastic setting or not, we cannot completely rely on Procedure 5.1 as the sole optimization engine – to repeat the above, the resulting algorithm would both be too slow and likely to get trapped in local maxima. We employ two ideas to avoid these difficulties.

I. The optimal scaling problem. Consider the following variation of the optimal control problem, assuming deterministic outages ($\epsilon_r = 0$ for all r). At round r, for each island K of G^r, all loads in K are scaled by a common value $0 \leq \lambda^r_K \leq 1$. That is to say, in Step 3 of Template 3.1 we reset

$$\text{new demand at } v \quad = \quad \lambda^r_K d^r_v \tag{12}$$

for each load bus v in K. The objective is to choose all the λ parameters so as to maximize demand being delivered at termination of the cascade.

At first glance, this problem might seem to be ill-posed, since potentially there are an exponential number of islands K that could be realized in different rounds, and in principle we need to specify λ parameters for each of these islands. However, a given control need only specify at most Rn nonzero values λ^r_K (here n = number of buses). This is seen as follows. At round $r = 1$ we only need to choose λ^1_K for those islands K that are actually in existence at the start of the cascade. Having chosen the λ^1_K, and since we use a deterministic outage rule, there will be a unique realization of islands at round 2, and consequently only the values λ^2_K corresponding to those islands need to be specified. This process continues inductively so that at any round, at most n multipliers need to be specified.

It is possible to show that rule (12) is a special case of the affine control law (7). Further, we have:

Theorem. In the memory-free version of outages ($\alpha = 1$ in eq. (2)), an optimal scaling control can be computed in time $O(m^R/R!)$.

We will skip the proof of this theorem for economy of space. As a result of the theorem, there is a theoretically efficient algorithm to solve the optimal scaling problem, and in fact, initial testing of the algorithm shows it to be far more efficient than the worst-case bound in the theorem. This is significant because it allows us to solve, *exactly*, a problem of the general type that we are interested in. Of course, the conditions under which the theorem applies are restrictive; even so however we expect that the application of the theorem should discover useful "structure" of the optimal controls that would carry across models.

Moreover, the theorem extends to other cases (such as imposing ranges on the allowable scale values, restricting control to a subset of the rounds, and other restrictions) and we estimate that the memory-free condition can be circumvented. A focus of forthcoming work will be to appropriately extend the theorem so as to obtain *robust* scaling controls.

Our initial experiments solving the scaling problem appear promising. By the time of the conference we hope to report on more substantial tests.

II. Segmented search. Consider a fixed partition $(\Sigma_1, \Sigma_2, \ldots, \Sigma_H)$ of the load buses, and consider triples $(\hat{c}^r_i, \hat{b}^r_i, \hat{s}^r_i)$ for each $1 \leq r < R$ and $1 \leq i \leq H$. We can use these triples to define an affine control (7): for each $1 \leq r < R$, and each demand bus v, we set $(c^r_v, b^r_v, s^r_v) = (\hat{c}^r_i, \hat{b}^r_i, \hat{s}^r_i)$ if $v \in \Sigma_i$.

An example (which we use in our implementation) is that where the Σ_i are *demand quantiles*. That is to say, if L is the number of demand buses, then Σ_1 contains the $\lfloor H/L \rfloor$ buses with largest demand, Σ_2 contains the next $\lfloor H/L \rfloor$ buses with largest demand, and so on.

The first-order algorithmic template 5.1 is easily adapted to a (segmented) first-order search. The advantage of this approach is that it considerably reduces the dimensionality of the problem, even if H is chosen relatively large, such as $H = 100$. Further, a segmented control is arguably 'fair' in that it specifies, to some degree, that similar buses are bound by similar control laws, though we stress that when applying our control the actual demand reduction can be very different for two buses in the same segment but in different islands.

VI. Second set of experiments

We report on a parallel implementation of the first-order method using the familiar master-worker paradigm. Each worker process performs computations of the yield function $\tilde{\Theta}^R(c, b, s)$ for controls (c, b, s) whereas the master carries out the gradient search algorithm. In the experiments we report on here, we use the linear power flow model; the resulting Laplacian systems are solved as (nominal) linear programs using Cplex 12.0 [14] and Gurobi 3.0 [15]. Interprocess communication uses Unix sockets. The computations below were performed on three eight-core i7 machines with 48GB of RAM each, for a total of 24 worker processes.

Our testing procedure is the same as in Section IV: we interdict the grid by removing K random lines, we then compute a control and finally we report on the behavior of the control. Table V reports on experiments using $R = 4$ rounds; when applying the memory rule (2) we used $\alpha = .55$, and we used the deterministic outage rule. The segmented first-order search was performed using $H = 50$ segments, and fixing $b_i^r = 1$ for all rounds r and segments i. In addition, each iteration of the first-order procedure Procedure 5.1 was broken up into two sub-iterations; in the first we fix the c values and perform a step restricted to the s-coordinates, and in the second the reverse. The step-size computation Step 2 in was carried out by first rescaling the gradient (to norm 1) and then evaluating 100 steps in multiples of 0.01.

TABLE V
Performance of algorithm on 4-round cascades

K	**yield, no control**	**yield, control**	**wallclock (sec)**
1	90.04	95.03	268
2	1.25	50.13	174
5	32.94	81.05	214
10	2.02	36.97	194
20	1.64	27.84	220
50	0.83	16.96	477

In Table V, 'wallclock' is the observed parallel running time of the first-order method, which dominates.

Note that in the case $K = 1$ the interdiction has limited effect, but even so the control is able to recover additional demand. In the case $K = 5$ the demand loss in the no-control case is substantial, but so is the benefit of the control. Finally, in the cases $K = 2, 10, 20, 50$ the network collapses but the control sill recovers a significant amount of demand. More experiments of this type will be forthcoming.

In the next set of experiments we use the case $K = 50$ in Table V to investigate in more detail the behavior of the algorithm as R increases. We used $\alpha = 0.5$ for all these experiments. Note that keeping α constant but increasing R effectively considers cascades that take longer from a 'real time' perspective, thereby giving more power to an agent applying control. If, instead, we were to increase R while also decreasing α, thus giving more weight to 'history', we would be able to model cascades that last for a fixed period of time, but where the individual rounds encompass shorter spans of time.

Table VI reports on the experiments. The column labeled 'grad steps' reports the number of gradient steps.

TABLE VI
Impact of increasing number of rounds

R	**yield no control**	**yield gradient**	**wallclock (secs)**	**grad steps**
5	4.13	31.86	1340	7
6	2.02	25.86	657	6
7	2.25	25.98	434	3
8	0.78	37.72	3151	10

Comments on Table VI:

Computational workload. Consider the case $R = 8$. Since we are using $H = 50$ segments, we have altogether 100 control variables c_i^r and s_i^r per round r. Since there are 7 rounds during which we will apply control, we have a total of 700 individual variables. Each partial derivative estimation requires two simulations; thus in total each gradient estimation entails 1400 cascade simulations. Per iteration, the step-size computations require 200 additional cascade simulations; for a total of 1600 simulations per iteration of Procedure 5.1. The case $R = 8$ required 10 gradient iterations, and thus in total 16000 simulations. Each 8-round simulation (of the 15000-bus Eastern Interconnect, and using one core of the i7 CPU) requires, on average, 4.5 CPU seconds. This is primarily due to the two power flow computations per round, and linear solver data structure cleanup at the end of the simulation (and to a much lesser degree, to graph algorithms used to identify islands). Thus in total the computation of the $R = 8$ case required approximately 72000 CPU seconds. Since we have 24 worker cores, this translates to approximately 3000 wallclock seconds. The balance of time with respect to the actual wallclock time in Table VI (i.e., 151 seconds) is due to inter-process communication and networking delays, and logging of statistics to disc by the master. On a per-simulation, per-core basis, this amounts to $151 * 24/16000 \approx 0.22$ seconds, or roughly 5% as compared to 4.5 seconds total per simulation.

Qualitative issues. A comparison of the entry in Table VI for $R = 5$ and those for $R = 6, 7$ might appear to indicate that the controls computed for $R = 6$ and 7 are locally optimal, because the control that achieves yield 31.86% for $R = 5$ "should be" feasible for all $R \geq 5$.

While it is true that the controls in Table VI can all be improved upon, the argument in the above paragraph is not quite correct. Refer to our Cascade Control template 3.1. The termination step constitutes a last-recourse form of control – if there are line overloads at the start of the last round, loads are scaled so as to remove the overloads, and in that case the cascade is considered terminated, regardless of history (and of particular, of rule (2). We model termination this way on purpose, so as to provide an agnostic termination criterion that does not depend on numerical parameters of our model, in particular, α. Consider Table VII.

TABLE VII
Maximum line overload at end of each round on case K = 50 case from Table VI

	C_5	C_6	C_7	C_8	**None**
1	6.47	1.83	2.22	3.79	177.83
2	14.12	1.83	1.57	33.49	122.06
3	36.79	1.23	1.30	6.90	114.45
4	1.72	1.14	2.26	6.70	22.47
5		0.99	1.18	59.33	45.43
6			1.08	1.98	40.33
7				1.18	114.90

In this table, the columns labeled "C_k", for $k = 5, \ldots, 8$ represent the controls in Table VI, whereas "None" means no control. The table shows, for each round, the maximum line overload at the *end* of that round, for each control option. We see that C_5 reaches the start of the termination round, round 5, with maximum overload 1.7232; the current yield at the start of round 5 is 54.90% (not shown in the Table) and most of the demand is in one island. Hence the termination step will scale demands by $1/1.7232$ and yield will drop to $54.90/1.7232 \approx$

31.86, as Table VI shows. As per our rules, this terminates the cascade, although since $\alpha = 0.5$, and because the end-or-round 3 maximum overload is very large, the maximum history-dependent line overload will be much larger than 1.732 (it should be at least $0.5 * 36.79 = 18.40$. Hence control C_5, if implemented in a cascade with 6 or more rounds, will not result in a stable state by the end of round 5.

Another point that emerges from Table VII is that C_5 and C_8 tend to maintain higher line overloads late into the cascade – this is a severe cascade, and having more time to apply control pays off. But by doing so C_5 and C_8 are likely less robust. Rather than performing the same robustness analysis as in Section IV, we will next consider the stability of the controls with respect to the α parameter in eq. (2) which in the above tests was set to 0.5.

This is a delicate issue, because the value of α is related to the time duration of a round, and thus the structure of an optimal control *should* depend on α (in other words, how much time we have impacts what kind of control we can apply). The question is how stable a control remains as α is perturbed.

TABLE VIII
Stability of controls in Table VI as a function of α

α	C_5	C_6	C_7	C_8
0.45	1.49	25.05	24.52	27.10
0.46	1.49	25.33	24.52	25.31
0.47	28.49	25.33	24.52	25.31
0.48	28.47	25.33	24.52	26.08
0.49	28.47	25.33	24.52	28.56
0.50	31.86	25.86	25.98	37.72
0.51	21.99	25.86	25.98	34.11
0.52	20.99	25.86	25.98	35.94
0.53	20.99	25.86	25.98	32.75
0.54	20.99	25.86	25.98	32.75
0.55	20.99	25.86	25.98	31.83

In Table VIII we show the yields obtained by running the C_k controls from Table VI using their respective numbers of rounds, but using different values for α. We see that in terms of the deviation from the nominal case (i.e., $\alpha = 0.5$), C_6 and C_7 prove the most stable, C_8 significantly less so and C_6 is very unstable. It is still the case that C_8 remains best overall: this is due to the severity of the cascade.

VII. Conclusion

A qualitative observation that emerges from the experiments in Sections IV and VI is that postponing control to later rounds can make a control strategy less robust. This is intuitively clear. On the other hand, an opposing observation that emerges is that in the case of a severe cascade it may not be best to try to stop the cascade immediately, to some degree allowing unavoidable outages to take place. A robust control takes advantage of the two observations.

In forthcoming work we plan to experiment with a robust version of the optimal scaling problem in Section V. We expect that this will provide an excellent start for the (stochastic gradients) algorithm 5.1. A second focus will be in adapting our machinery to AC power flows.

Acknowledgment

We would like to thank Ian Dobson and Ian Hiskens for fruitful discussions, and for making the Eastern Interconnect data available to us.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall, NJ (1993).
[2] R. Alvarez, Interdicting Electric Power Grids, Masters' Thesis, U.S. Naval Postgraduate School, 2004.
[3] G. Andersson, *Modelling and Analysis of Electric Power Systems*. Lecture 227-0526-00, Power Systems Laboratory, ETH Zürich, March 2004.
[4] A. Bergen and V. Vittal, *Power Systems Analysis*, Prentice-Hall (1999).
[5] D. Bienstock and S. Mattia, "Using mixed-integer programming to solve power grid blackout problems," *Discrete Optimization* Vol. 4 (2007), 115–141.
[6] D. Bienstock and A. Verma, "The $N - k$ Problem in Power Grids: New Models, Formulations, and Numerical Experiments," *SIAM J. Opt.* Vol 20 (2010), 2352–2380.
[7] V.M. Bier, E.R. Gratz, N.J. Haphuriwat, W. Magua, K.R. Wierzbickiby, "Methodology for identifying near-optimal interdiction strategies for a power transmission system, " Reliability Engineering and System Safety Vol 92 (2007), 1155–1161.
[8] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press.
[9] B.A. Carreras, V.E. Lynch, I. Dobson, D.E. Newman, "Critical points and transitions in an electric power transmission model for cascading failure blackouts," Chaos, vol. 12, no. 4, 2002, 985-994.
[10] B.A. Carreras, V.E. Lynch, D.E. Newman, I. Dobson, "Blackout mitigation assessment in power transmission systems, " 36th Hawaii International Conference on System Sciences, Hawaii, 2003.
[11] B.A. Carreras, V.E. Lynch, I. Dobson, D.E. Newman, "Complex dynamics of blackouts in power transmission systems," Chaos, vol. 14, no. 3, September 2004, 643-652.
[12] B.A. Carreras, D.E. Newman, I. Dobson, A.B. Poole, "Evidence for self organized criticality in electric power system blackouts," IEEE Transactions on Circuits and Systems I, vol. 51, no. 9, Sept. 2004, 1733-1740.
[13] A.R. Conn, K. Scheinberg and L.N. Vicente, *Introduction to derivative-free optimization*, MPS-SIAM Series on Optimization, Philadephia (2009)
[14] IBM ILOG, Incline Village NV.
[15] Gurobi Optimization, Houston TX.
[16] J.T. Linderoth and S. J. Wright, "Decomposition Algorithms for Stochastic Programming on a Computational Grid, " *Computational Optimization and Applications* Vol 24 (2003) 207–250.
[17] H.J. Kushner and D.S. Clark, *Stochastic approximation methods for constrained and unconstrained systems.* Springer-Verlag Berlin, (1978).
[18] D.G. Luenberger, *Linear and Nonlinear Programming*, Addison-Wesley (1984).
[19] A. Pinar, J. Meza, V. Donde, and B. Lesieutre, "Optimization Strategies for the Vulnerability Analysis of the Electric Power Grid," *SIAM Journal on Optimization* Vol 20 (2009), 1786–1810.
[20] H. Robbins and S. Monro, "On a stochastic approximation method," *Annals of Mathematical Statistics* Vol 22 (1951), 400 - 407.
[21] *Final Report on the August 14, 2003 Blackout in the United States and Canada: Causes and Recommendations*, U.S.-Canada Power System Outage Task Force, April 5, 2004. Download from: https://reports.energy.gov.
[22] A. Wächter and L. T. Biegler, "On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming," Mathematical Programming Vol 106 (2006), 25 – 57.

Daniel Bienstock Daniel Bienstock works on computational applied mathematics with focus on methodology of large-scale optimization and applications to engineering, physics and biology. He received the PhD in Operations Research from MIT in 1985.