De-risking solutions to optimization problems*

Daniel Bienstock and Blake Sisson, Columbia University

June 29, 2026

Abstract

We develop a cutting-plane methodology that adjusts solutions to optimization problems so as to reduce features that bring about exposure to risk, such as concentration of assets or resources. The methodology is agnostic to the representation of risk but has provably good attributes. Our procedure aims to reduce the appropriate risk metric without accruing a significant increase in nominal cost, rapidly, or proves that such an adjustment is not possible. The underlying approach borrows from techniques used in first-order methods for optimization.

1 Introduction

Consider a generic optimization problem

$$\min c(x), \text{ s.t. } x \in P. \tag{1}$$

An empirical fact which is found in many real-world examples is that optimal or near-optimal solutions end up, inadvertently, incorporating high *concentration*. For example, in a logistical setting, a solution may place a large percentage of high-value items in a small set of locations during a narrow time span, causing exposure to many different types of actual risk. In a sense, concentration acts as an enabler for exogenous risk. This notion of concentration risk is well-known in multiple industrial settings. Notably, a concrete algebraic representation of concentration suitable for optimization may be very high dimensional.

In this paper we will assume that concentration (or, when appropriate, risk) is approximately quantified through a function $\Phi(x)$. To fix ideas we will refer to $\Phi(x)$ as the *impact* function. The algorithms discussed in this paper efficiently probe the frontier defined by cost versus impact, in order to rapidly de-risk an optimal solution x^* to (1) by either

(a) computing an alternative vector $\hat{x} \in P$ with *significantly lower impact but moderate increase in cost* relative to x^*, or

(b) proving that no such vector exists.

Thus, in simple terms, we develop computational tools that support an *optimistic* risk-tolerance stance, when possible, and that yield at least a partial explanation (i.e., an impossibility proof) when not achievable.

*Funded by AFOSR and ONR.

Preliminary version: Wed.June.30.165033.2026

1.1 Motivation

There is a large literature pointing to undesirability of concentration arising from optimization in varied settings such as finance and logistics. See, e.g., [26][1]. Here we present an example to fix ideas.

The plot in Figure 1 was produced by solving a logistical MIP that models shipping a variety of commodities using several vehicle types on a medium-sized network in a multiperiod setting. The model is endowed with several types of capacities, vehicle availabilities, deadlines and vehichle-commodity compatibility rules. The particular case considered here has over 400,000 variables of which over 5,000 are integral.

Figure 1 displays sorted *activity* weights – an activity is the set of all shipments on a given network link at a given point in time, over all available vehicles. Even though there are over 10,000 such activities, only approximately 150 are nonzero at the optimum. More significantly, we see very high concentration near the top.

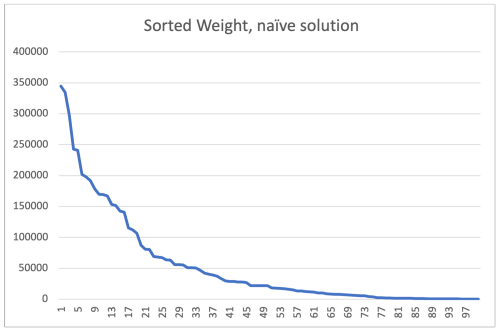

Figure 1: Sorted activity weights.

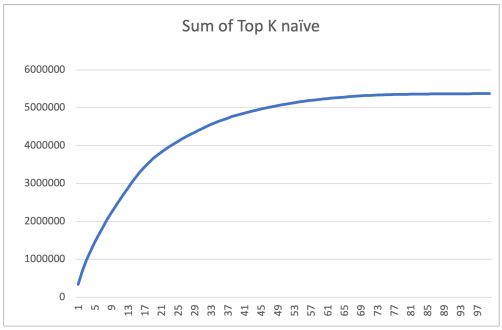

Figure 2: Aggregate weight of top activities.

Figure 2 plots the aggregate weight of the top K activities, for $K = 1, 2, \ldots$ Note that, for example, the top ten activities account for approximately 40% of all shipped weight. Such concentration is risk-inducing in an intuitive fashion: congestion or an adversarial event that affects a small number of activities will have a negative impact out of proportion to value of the concentration.

As a simple example of the computational challenges inherent in addressing concentration risk, consider a logistical setting where there are decision variables of the form $x_{j,t}$, modeling commodity levels present at location j at time t. Suppose we define

$$\Phi(x) \doteq \max_{H \in \mathcal{J}} \sum_{t \in H} \sum_{j \in \mathcal{S}_H} x_{j,t} \tag{2}$$

where $\mathcal{J}$ is a family of (short) time intervals and for each interval H, $\mathcal{S}_H$ is a set of locations, possibly dependent on x itself. An example is provided by the "top K" setup ($K = 2, 3, \ldots$, small) and for $H \in \mathcal{J}$, $\mathcal{S}_H$ is the set of K highest values $x_{j,t}$ among all locations j and all time periods t within H. In that case, (2) takes the form

$$\Phi(x) \doteq \max_{H \in \mathcal{J}} \sum_{t \in H} \sum_{k=1}^{K} x_{(k),t} \tag{3}$$

[1] In particular see page 13, where high dimensionality and opacity of risk are brought up. "For example, having diversified supply sources for a particular part of the supply chain might not mean much if the disruption is from somewhere unexpected. Companies do not know what they do not know."

where for $1 \leq k \leq K$, $x_{(k),t}$ is the k^{th} largest value $x_{j,t}$ with $t \in H$. Thus, $\Phi(x)$ is large when there is high concentration of commodities in a narrow time frame at a small set of locations. A computational challenge arises in that an explicit representation of $\Phi(x)$ will require a high dimensional formulation. Additionally, rather than using an aggregate quantity $x_{j,t}$ it may prove useful to refer to specific commodities, further increasing dimensionality. Incorporating a model of exogenous uncertainty into $\Phi(x)$ will further extend the dimensionality.

For a very different example consider an optimization problem

$$\min f_0(x) \quad \text{s.t.} \sum_{j \in J_i} a_{ij} f_{ij}(x) \ \leq \ b_i, \ \forall i \in I, \ x \in \mathbf{R}^n, \tag{4}$$

where $a_{ij} \neq 0$ and $f_{ij}(x)$ is a given function (e.g., a monomial) for each $i \in I$ and $j \in J_i$. Suppose the a_{ij} are uncertain, under a model where a_{ij} is replaced by $a_{ij}(1+z_{ij})$ with z is chosen from within some set $\mathcal{Z}$. We can thus define

$$\Phi(x) \doteq \max_{z \in \mathcal{Z}} \max_{i \in I} \left\| \left[\sum_{j \in J_i} a_{ij} f_{ij}(x)(1+z_{ij}) - b_i \right]^+ \right\|_p \tag{5}$$

for some $p \geq 0$ as a measure of the adversarial infeasibility incurred by x feasible for (4). Our algorithms seek to compute x that is near-optimal for (4) but that achieves (much) smaller maximum infeasibility error than a nominally optimal solution to (4).

1.1.1 Discussion of a disadvantageous approach

Our de-risking goal can, in principle, be addressed by solving formulations of the form

$$\min \ c(x) \ \text{ s.t. } \ x \in P, \quad \Phi(x) \leq \Lambda, \tag{6}$$

where Λ is an input value that explicitly states a decision-maker's bound on risk. We next argue that this is an unfavorable methodology.

1. Nontrivial selection for Λ. A priori, it may not be clear what value Λ should take, and why. Any choice for Λ reflects a risk-tolerance stance, and, ideally, one should estimate an efficient frontier by solving (6) for multiple values Λ. Additionally, the solution to (6) (vector and value) may be very sensitive to the choice of Λ especially in nonlinear, nonconvex cases. At any rate, it is the corrective goal mentioned above; de-risking an optimal solution to (1), that is of interest to us, and not a specific choice of an impact upper-bound Λ. Our de-risking goal is explicitly imprecise but allows for an *approximate* and implicit choice for Λ that is supported by a rigorous argument. See Lemma 2.10.

2. Computational challenges arising from an explicit choice for Λ. An additional hazard arises from a computational standpoint: a choice that is aggressive –Λ is too small– would cause problem (6) to become infeasible or nearly so [2]. In general, especially in the nonlinear and nonconvex case, a specific choice for Λ can give rise to solution times that are unacceptably lengthy. In fact the solver might simply fail, while, potentially, a slightly different choice yields a much faster solution.

3. An overly complex and large formulation. As highlighted in the examples underlying equations (3) and (4), the explicit representation of $\Phi(x)$ in (6) might require a formulation of great complexity

[2] In fact the computation of the smallest value Λ such that (6) remains feasible can be a nontrivial problem even in cases where (1) is a linear: it includes the min-cut problem as a special case.

and unacceptably large size, causing problem (6) to become far more challenging than (1). In both cases $\Phi(x)$ is of the form $\max_{i\in I}\phi_i(x)$ where I is a very large set and the $\phi_i(x)$ functions are nonlinear and require a description which is, itself, nontrivial, even in a purely deterministic setting. If, in addition, we have an exogenous uncertain component, the challenge becomes even more pressing.

1.1.2 Our approach

We develop cutting-plane algorithms that aim to parsimoniously de-risk an optimal solution x^* to the nominal problem (1), i.e., to compute $\hat{x} \in P$ with small increase in cost together with large decrease in impact, both relative to x^*, or to prove that this goal is not possible. While such de-risking cannot be guaranteed (because of structure of the feasible set P), experiments indicate that it is frequently achievable in relevant cases.

Our procedures are motivated by the impact-weighted formulation

$$\min\ c(x) + \Theta\Phi(x), \quad \text{s.t. } x \in P, \tag{7}$$

where $\Theta > 0$ is a risk-aversion parameter. Of note,

(a) A typical run of the procedures will *not* solve (7) and will be truncated short of that goal.

(b) We provide theoretical guarantees for our de-risking performance.

The value Θ must, of course, be selected: we will provide a method to resolve this issue that is provably correct. To a partial extent, a choice for Θ implies an equivalent choice for Λ in (6) – however, this equivalence only applies to formally optimal solutions to the respective problems. This equivalence is bypassed by (a) and formulation (7) proves more flexible than (6) when an approximate answer is desired.

In the above descriptions, the impact function $\Phi(x)$ is presented in abstract form. Potentially, this includes an adversarial ingredient. As an example in the logistical setting given above, consider a family of joint distributions $\mathcal{Z}$, each of which describes an uncertain event that results in capacity loss in network links. The risk-aligned quantity of interest, here, would be a measure of the amount of worth of flow that is delayed or lost under distribution $z \in \mathcal{Z}$. We will expand on these issues below.

This paper is organized as follows. Section 1.2 describes related work and Section 2.1 presents our methodological contributions which are analyzed in Section 2.3 (in particular, see Theorem 2.8). Section 2.4 presents simple applications of our algorithms, with more substantial experiments given in Section 3.

1.2 Review of prior work

The perspective we take in this paper concerns risk arising from solutions to optimization problems that incorporate concentration, in a generic sense. One of the earliest citations in this context is [18], which concerned routing problems (in packet networks) that can formally be stated as linear optimization problems. However, congestion brings about the risk of queueing delays: let the flow on an arc with capacity u be x: then the M/M/1 queueing delay is $u/(u-x)$ (see, e.g., [25]). The combinatorial algorithm in [18] approximately solves an optimization problem whose objective function is the original cost function, plus the sum of all queueing delays, thus trading-off cost versus queueing risk (i.e., delays).

A related problem was later taken up in [32]. They consider a capacity-constrained multi-commodity flow routing problem, where the capacities are insufficient to route all commodities.

Under such conditions, the algorithm in [32] computes a routing that approximately minimizes the maximum overload in any arc, which is defined as the ratio of flow to capacity. The algorithm replaces this min-max task, with an approximately equivalent nonlinear optimization problem which is (again, approximately) handled using a first-order method.

This approach was greatly extended and improved in [30], and [20]. Let $\mathcal{A}$ denote the set of arcs in a multicommodity flow network, and for an arc $(i,j) \in \mathcal{A}$, let u_{ij} denote its capacity, and for a (multicommodity) flow vector f let f_{ij} denote the sum of commodity flows routed on (i,j). Then the min-max routing problem described in the above paragraph can be stated as

$$\min_{f \in \mathcal{F}} \max_{(i,j) \in \mathcal{A}} f_{ij}/u_{ij}, \tag{8}$$

where $\mathcal{F}$ is the set of feasible flows, i.e., all flows that deliver all commodities from their source to their destination, possibly exceeding capacities. On very large networks with many commodities this optimization problem, which can be stated as a linear program, proves quite challenging to modern LP solvers. In contrast, [30] and [20] show that the min-max problem can approximately be modeled using an appropriate nonlinear *potential function* as a proxy:

$$\min_{f \in \mathcal{F}} \ln \left(\sum_{(i,j) \in \mathcal{A}} e^{\alpha f_{ij}/u_{ij}} \right). \tag{9}$$

More precisely, given $0 < \varepsilon < 1$, choosing α appropriately (primarily, α proportional to ε^{-1}) a solution to the potential function problem (9) yields a flow vector f that is ε-optimal to the min-max problem (8). Moreover, the approximate minimization in (9) is carried out, effectively, via a first-order method where individual iterations amount to linear programs whose objective is the gradient of the function in (9). We will rely on a similar proxy in our work, though our algorithmic goals and methodology are quite different.

As a function of ε, the number of iterations needed to obtain an ε-optimal solution to (8) grows proportional to $1/\varepsilon^2$ (later improved to $1/\varepsilon$, [10]). See [9] for an analysis of [18], [30] and [20] and follow-up work that improved on these algorithms, as well as computational experiments.

Our risk perspective can include an exogenous adversarial component. There is a very large literature of models for *interdiction* that take an adversarial standpoint: how to optimally or maximally disrupt, e.g., a logistical operation. In this regard, the case of a network under stochastic capacity interdiction is studied in [17] (also see references therein). This work considers a variety of models where an adversary interdicts arcs of a network by reducing their capacity under various assumptions of likelihood of success of interdiction and of capacity reduction, as well as adversarial capabilities; and produces efficient formulations for such problems. Optimization problems where under the risk of adversarial interdiction are often handled using a cutting-plane algorithm that is reminiscent of (or equivalent to) Benders' decomposition [4]. Related problems are considered in [12].

A salient and challenging application of interdiction analysis involves the so-called $N-K$ problem in power grids; see [31], [11]. Briefly, an operational constraint in modern power grids is that *any* simultaneous disablement of up to K components (e.g., arcs) should be tolerated by the system; here $K > 0$ is a small integer. The case $K = 1$ is enshrined as law in many countries, but larger values of K are becoming relevant. Note that the enumeration of all subsets of size K is infeasible even for relatively small K (e.g., 3) when the grid is large, and a generic Benders' approach, when applicable, proves invaluable.

Optimization problems under interdiction risk are usually (if unwittingly) modeled as bilevel optimization problems, for which there is also a very abundant literature. Bilevel optimization

problems are complex and Benders' decomposition is not always an easy or even feasible task. For a recent survey, see [3].

Finally, the incorporation of an adversarial model can bring about the robust optimization perspective, especially in the so-called "risk-budgets" setup. See [6] and [5]. Many of the previously cited works on interdiction models can be seen through the lens of risk budgets.

2 Model and algorithms

We describe cutting-plane algorithms for approximately solving problem (7) in a setting where the function $\Phi(x)$ captures risk due to concentration with high-dimensional features. The problem setup we will consider has the following characteristics.

As inputs, we are given a compact set $\mathcal{Z}$ that parameterizes uncertainty, and for each $z \in \mathcal{Z}$,

(a) A set I of nonnegative-valued functions of the form $\phi_i(x|z)$ ($i \in I$). We assume that I is finite though potentially very large. We will refer to the ϕ_i as the *features*. Each feature is a function of the decision vector x as well as the exogenous risk-inducing vector $z \in \mathcal{Z}$.

(b) A set of values $\lambda_1 \geq \ldots \geq \lambda_{|I|} \geq 0$ with $\sum_{i=1}^{|I|} \lambda_i = 1$. For $x \in P$ we define the impact on x of event z

$$\Phi(x|z) \doteq \sum_{k=1}^{|I|} \lambda_k \phi_{(k)}(x\,|\,z),$$

where $\phi_{(k)}(x\,|\,z)$ is the k^{th} largest value $\phi_i(x\,|\,z)$. We set

$$\Phi(x) \doteq \max_{z\in\mathcal{Z}} \Phi(x|z).$$

Thus problem (7), in its epigraph representation, takes the form

$$\min\ c(x) + \Theta\Phi_L \tag{10a}$$

$$\text{s.t. } x \in P, \quad \Phi_L \geq \max_{z\in\mathcal{Z}} \sum_{k=1}^{|I|} \lambda_k \phi_{(k)}(x\,|\,z). \tag{10b}$$

Denote by $\mathcal{S}_{|I|}$ the set of permutations of $|I|$ entities, and for $\pi \in \mathcal{S}_{|I|}$ (with a slight abuse of notation) use $\pi(i)$ to denote the corresponding element i of I. Thus, for $x \in P$ and $z \in \mathcal{Z}$, we have

$$\sum_{k=1}^{|I|} \lambda_k \phi_{(k)}(x\,|\,z) = \max_{\pi\in\mathcal{S}_{|I|}} \sum_{k=1}^{|I|} \lambda_k \phi_{\pi(k)}(x\,|\,z). \tag{11}$$

In particular, when $\lambda_k = 0$ for $k > 1$ problem (10) becomes

$$\min\ c(x) + \Theta\Phi_L \tag{12a}$$

$$\text{s.t. } x \in P, \quad \Phi_L \geq \max_{z\in\mathcal{Z}} \max_{i\in I} \phi_i(x\,|\,z). \tag{12b}$$

Using (11), we note that (10) is, in fact, a *special case* of (12) with appropriately redefined features (e.g., one for each $\pi \in \mathcal{S}_I$). *For simplicity of language*, in the theoretical results below we assume the form (12), though our computational experiments include cases with the more general form.

An example of this setup is that where each function $\phi_i(x|z)$ describes a stochastic risk measure (e.g., a conditional value-at-risk), and $\mathcal{Z}$ represents a set of realizations of uncertainty.

Preliminary version: Wed.June.30.165033.2026

2.1 Prototype algorithm

Here we describe a procedure, Algorithm 1, which has provable convergence attributes. Notably, this procedure does not necessarily solve problem (12) to optimality, however it provably produces an output vector with desirable attributes.

This algorithm serves as a common intellectual core for a family of procedures that prove empirically successful. These procedures, which are discussed in Section 2.3.3, incorporate pragmatic modifications to Algorithm 1 for high-dimensional cases and in some cases are also endowed with provably correct attributes.

Algorithm 1 SOFTMAX-ADVERSARIAL

1: **Inputs:** $t^{\max} > 0$ (max. no. of iterations), $\Theta > 0$; $\alpha > 0$; $\Delta, \delta < 1$, Δ' (positive tolerances).
2: Set $t = 0$, and initialize the *master problem* as

$$\min\ c(x) + \Theta\Phi_L, \quad \text{s.t. } x \in P,\ \ \Phi_L \geq 0.$$

3: **while** $t < t^{\max}$ **do**:
4: Solve the master problem; let (x^t, Φ_L^t) be an optimal solution.
5: **Boosting step.** Compute $z^t \in \mathcal{Z}$ such that

$$\ln \sum_{i\in I} e^{\alpha\phi_i(x^t|z^t)} \ \geq\ \max_{z\in\mathcal{Z}} \ln \sum_{i\in I} e^{\alpha\phi_i(x^t|z)} \quad - \quad \Delta'.$$

Define $\phi^t_{\max} = \max_{i\in I} \phi_i(x^t|z^t)$.
6: **if** (a) $\phi^t_{\max} \leq \Phi^t_L + \Delta$, or (b) $\phi^t_{\max} - \Phi^t_L \leq \delta\, \phi^t_{\max}$ **then STOP**.
7: **else**
8: For each $i \in I$ write $\pi^t_i = \frac{e^{\alpha\phi_i(x^t|z^t)}}{\sum_{j\in I} e^{\alpha\phi_j(x^t|z^t)}}$
9: **Separation.** Add the cut $\Phi_L \geq \sum_{i\in I} \pi^t_i\, \phi_i(x|z^t)$ to the master problem.
10: **end if**
11: $t \leftarrow t + 1$.
12: **end while**

2.2 Preliminary discussion of Algorithm 1

For $x \in X$ recall that $\Phi(x) = \max_{z\in\mathcal{Z}} \max_{i\in I} \phi_i(x|z)$. Consider any iteration $t \geq 0$. Note that $\pi^t \geq 0$ and $\sum_i \pi^t_i = 1$. As we will see below (Lemma 2.1), this implies $\Phi^t_L \leq \Phi(x^t)$, and, as a result, the master problem is always a relaxation of (12). Moreover,

- $\Phi(x^t) - \Phi^t_L$ amounts to the infeasibility of (x^t, Φ^t_L) in problem (12) – should this quantity be small, the algorithm will have converged to an approximate optimal solution.
- By definition $\phi^t_{\max} \doteq \max_{i\in I} \phi_i(x^t|z^t) \leq \Phi(x^t)$,
- Furthermore (Lemma 2.2), $\Phi(x^t) \leq \phi^t_{\max} + 2\Delta$.

In summary, $\phi^t_{\max}$ is a valid proxy to $\Phi(x^t)$, and, as a result, the stopping conditions in line 6 are numerically valid termination conditions for the cutting plane algorithm (absolute and relative, resp.). In order for this statement to be mathematically valid we need that δ be small in absolute terms, e.g., $\delta = 10^{-6}$, and that Δ be small be small in relative terms, i.e. $\Delta \leq 10^{-6}\phi^t_{\max}$. In the numerical

implemenation of Algorithm (1) we will relax these conditions in order to allow early termination, in line with our 'de-risking' goal.

Remark. In machine learning language, the function optimized in line 5 of the algorithm is known as *log-sum-exp*, while the cut coefficients in line 8 follow the *softmax* function. Formally, given $y \in \mathbf{R}^n$, and $\alpha > 0$ we write $\text{SOFTMAX}(\alpha, y)_i \doteq \frac{e^{\alpha y_i}}{\sum_{j=1}^n e^{\alpha y_j}}$. It is seen that the softmax function is proportional to the gradient of log-sum-exp (in feature space), that is to say, defining $\phi = (\phi_i, i \in I)$, $L(\phi) = \ln \sum_{i \in I} e^{\alpha \phi_i}$ and $g(\phi)$ as the vector with entry $g_i(\phi) = \alpha \frac{e^{\alpha \phi_i}}{\sum_{j \in I} e^{\alpha \phi_j}}$ for each $i \in I$, we have $\nabla_\phi L(\phi) = g(\phi)$.

As defined, L is a convex function of ϕ. Thus, given a particular vector $\bar{\phi}$ it holds that

$$L(\phi) \geq g^T(\bar{\phi})(\phi - \bar{\phi}) + L(\bar{\phi}). \tag{13}$$

Using this observation, one can write a modification to Algorithm 1 similar to Kelley's [23] algorithm using the outer-approximation cuts (13) instead of the cuts added in line 9.

2.3 Analysis of Algorithm 1

We next present results regarding the convergence properties of Algorithm 1, concluding with Theorem 2.8 and Corollary 2.9. We will set $\Delta' = \alpha\Delta$ with $\alpha \geq \frac{\ln|I| + [\ln(2\Phi(x^0)/\Delta)]^+}{\Delta}$, which is well-defined since α is not actually needed until the first execution of line 5, i.e., not until after x^0 is computed.

Lemma 2.1 *At any iteration t, $\Phi_L^t \leq \Phi(x^t)$.*

PROOF: This is true for $t = 0$ since $\phi_L^0 = 0$, and, for $t > 0$, since $\Theta > 0$ we have that $\phi_L^t = \sum_{i \in I} \pi_i^{t'} \phi_i(x^t | z^{t'})$ for some $t' < t$. Since the $\pi_i^{t'}$ are nonnegative and sum to 1 the proof is complete. ■

Lemma 2.2 *At any iteration t, $\Phi(x^t) \leq \phi_{\max}^t + 2\Delta$.*

PROOF: Given $z \in \mathcal{Z}$, $e^{\alpha \max_{i \in I} \phi_i(x^t|z)} \leq \sum_{i \in I} e^{\alpha \phi_i(x^t|z)} \leq |I| e^{\alpha \max_{i \in I} \phi_i(x^t|z)}$ and therefore

$$\alpha \max_{i \in I} \phi_i(x^t|z) \quad \leq \quad \ln \sum_{i \in I} e^{\alpha \phi_i(x^t|z)} \quad \leq \quad \ln|I| + \alpha \max_{i \in I} \phi_i(x^t|z). \tag{14}$$

So

$$\alpha \Phi(x^t) \leq \max_{z \in \mathcal{Z}} \ln \sum_{i \in I} e^{\alpha \phi_i(x^t|z)} \leq \ln \sum_{i \in I} e^{\alpha \phi_i(x^t|z^t)} + \Delta',$$

as per line 5 of the algorithm. Using (14) with $z = z^t$, the rightmost quantity is at most $\ln|I| + \Delta' + \alpha \phi_{\max}^t$. In summary, $\Phi(x^t) \leq \frac{\ln|I| + \Delta'}{\alpha} + \phi_{\max}^t \leq \phi_{\max}^t + 2\Delta$ by definition of α and Δ'. ■

Corollary 2.3 *Suppose the algorithm terminates at iteration t (line 6). If criterion (a) applies, $\Phi(x^t) - 3\Delta \leq \Phi_L^t \leq \Phi(x^t)$. And if (b) applies $(1-\delta)\Phi(x^t) - 2\Delta \leq \Phi_L^t \leq \Phi(x^t)$.*

PROOF: This follows from the termination condition, Lemmas 2.1 and 2.2. ■

Observation 2.4 *Given t, let $\tilde{x} \in P$ be such that $c(\tilde{x}) < c(x^t)$. Then $\Phi(\tilde{x}) > \Phi_L^t$.*

This follows because $(\tilde{x}, \Phi(\tilde{x}))$ is feasible for the master problem at iteration t. In light of Corollary 2.3, this shows that even prior to termination, the algorithm yields useful information.

Lemma 2.5 *For any t, $\Phi_L^t \le \Phi(x^0)$.*

PROOF: For any $x \in P$, the pair $(x, \Phi(x)))$ is feasible for the master problem at iteration t. Using this fact with $x = x^0$, we get $c(x^t) + \Theta\Phi_L^t \le c(x^0) + \Theta\Phi(x^0)$. Since by construction $c(x^0) \le c(x^t)$, and $\Theta > 0$, the proof is complete. ■

In the proofs that follow, for a given t we define

$$S^t = \{i : \phi_i(x^t|z^t) \le \Phi_L^t\} \quad \text{and} \quad B^t = \{i : \phi_i(x^t|z^t) > \Phi_L^t\}.$$

Lemma 2.6 *Suppose that at iteration t the algorithm does not stop on line 6. Then the cut on line 9 is violated by (x^t, Φ_L^t) by more than $\frac{\sum_{i\in B^t} e^{\alpha\phi_i(x^t|z^t)}}{\sum_{i\in I} e^{\alpha\phi_i(x^t|z^t)}}\Delta/2$.*

PROOF: It suffices to show that

$$\sum_{i\in I} e^{\alpha\phi_i(x^t|z^t)}\Phi_L^t + \sum_{i\in B^t} e^{\alpha\phi_i(x^t|z^t)}\Delta/2 < \sum_{i\in B^t} e^{\alpha\phi_i(x^t|z^t)}\phi_i(x^t|z^t), \quad \text{i.e., that} \tag{15}$$

$$\sum_{i\in S^t} e^{\alpha\phi_i(x^t|z^t)}\Phi_L^t < \sum_{i\in B^t} e^{\alpha\phi_i(x^t|z^t)}(\phi_i(x^t|z^t) - \Phi_L^t - \Delta/2). \tag{16}$$

The left-hand side is upper bounded by $|S_t|e^{\alpha\Phi_L^t}\Phi_L^t < |S_t|e^{\alpha(\phi_{\max}^t - \Delta)}\Phi_L^t$, while the right-hand side is lower bounded by $e^{\alpha\phi_{\max}^t}(\phi_{\max}^t - \Phi_L^t - \Delta/2) > e^{\alpha\phi_{\max}^t}\Delta/2$. Hence the result follows if we can argue that

$$|S_t|\Phi_L^t < e^{\alpha\Delta}\Delta/2.$$

Using $|S| \le |I|$ and Lemma 2.5, this holds since $\ln|I| + \ln\Phi(x^0) \le \alpha\Delta + \ln(\Delta/2)$, by choice of α. ■

Corollary 2.7 *Under the same assumptions as Lemma 2.6, the violation of the cut added in line 9 is at least $\Delta/4$.*

PROOF: Consider the quantity $r \doteq \frac{\sum_{i\in B^t} e^{\alpha\phi_i(x^t|z^t)}}{\sum_{i\in I} e^{\alpha\phi_i(x^t|z^t)}}$. At least one of the terms in the numerator equals $e^{\alpha\phi_{\max}^t}$. If $|B^t| = 1$ then $r \ge \frac{e^{\alpha\phi_{\max}^t}}{e^{\alpha\phi_{\max}^t} + |S_t|e^{\alpha\Phi_L^t}} > \frac{1}{1+|I|e^{-\alpha\Delta}}$ (because $\Phi_L^t < \phi_{\max}^t - \Delta$) and, by choice of α, this ratio is larger than $1/2$ which concludes the proof. Now assume $|B^t| \ge 2$, and let p denote the *smallest* term in $\sum_{i\in B^t} e^{\alpha\phi_i(x^t|z^t)}$. In other words, we can write $\sum_{i\in B^t} e^{\alpha\phi_i(x^t|z^t)} = p + e^{\alpha\phi_{\max}^t} + q$, with $p, q \ge 0$, and so

$$r \ge \frac{p + e^{\alpha\phi_{\max}^t} + q}{p + e^{\alpha\phi_{\max}^t} + q + |S_t|e^{\alpha\Phi_L^t}}.$$

The numerator of the derivative of this expression with respect to p equals $|S_t|e^{\alpha\Phi_L^t}$ which is nonnegative, and so the expression is minimized when $p = 0$ and likewise with q. Thus, again, $r > \frac{1}{1+|I|e^{-\alpha\Delta}}$, as desired. ■

Under appropriate conditions, Corollary 2.7 implies finite termination of Algorithm 1. At any rate, this issue is addressed by the following result requires uniform continuity of the functions ϕ_i over Z. That is, for any $\varepsilon > 0$ there exists a $\gamma(\varepsilon) > 0$ such that, for any pair $z, w \in Z$, $\|z - w\| < \gamma(\varepsilon)$ implies $|\phi_i(x|z) - \phi_i(x|w)| < \varepsilon$ for every $x \in P$ and any $i \in I$.

Theorem 2.8 *Assume uniform continuity of the ϕ_i holds and let $\lambda \in (0,1)$. Either Algorithm 1 terminates finitely in line 6, or we reach an iteration τ with $\phi_{\max}^\tau \le \lambda\Phi(x^0)$.*

PROOF: Let $n = |I|$ and define $N > \max\{n, 2/\sqrt{\delta}\}$. Choose

$$\eta = \frac{\delta/2 - 2/N^2}{1 - 2/N^2}, \text{ and } \alpha \geq \frac{3\log N}{\eta\Delta}.$$

Further,

- Define $\varepsilon \doteq \dfrac{\lambda\phi_U^0\delta}{4}$.
- Let $\gamma(\varepsilon) > 0$ such that the uniform continuity of functions ϕ_i over Z holds for ε. Define $Z_{\gamma(\varepsilon)} \subseteq Z$ as a $\gamma(\varepsilon)$-net for Z, i.e. for each $z \in Z$, there exists a $z' \in Z_{\gamma(\varepsilon)}$ such that $\|z - z'\| \leq \gamma(\varepsilon)$. $Z_{\gamma(\varepsilon)}$ (finite) is guaranteed to exist since Z is compact.

At each iteration t, define $\hat{z}^t$ and $\hat{\pi}^t$ as follows:

- Let $\hat{z}^t \in Z_{\gamma(\varepsilon)}$ be the point whose $\gamma(\varepsilon)$-ball covers z^t.
- For each j, let $\hat{\pi}_j^t$ be π_j^t rounded down to the nearest integer multiple of $1/N^3$.

For any t, we say that index i is *minor* if $\phi_i(x^t|z^t) \leq (1-\eta)\phi_{\max}^t$, and *major* otherwise. By our choice of α, and since $\Delta < \phi_{\max}^t$ for every non-terminal iteration t, we have $e^{\alpha\phi_{\max}^t} > N^3 e^{(1-\eta)\alpha\phi_{\max}^t}$. Using this bound, and denoting $\pi_{\max}^t = \max_{1\leq i\leq n} \pi_i^t$, we see that for all minor i, $\pi_i^t < \frac{1}{N^3}\pi_{\max}^t$. So

$$\sum_{i \text{ minor}} \pi_i^t < \frac{1}{N^2}\pi_{\max}^t \leq \frac{1}{N^2} \sum_{j \text{ major}} \pi_j^t.$$

Hence, since $\sum_j \pi_j^t = 1$,

$$\sum_{j \text{ major}} \pi_j^t > 1 - \frac{1}{N^2} \text{ and thus } \sum_{j \text{ major}} \hat{\pi}_j^t > 1 - \frac{2}{N^2}, \text{ and therefore}$$

$$\sum_{j \text{ major}} \hat{\pi}_j^t \phi_j(x^t|\hat{z}^t) \geq \sum_{j \text{ major}} \hat{\pi}_j^t \phi_j(x^t|z^t) - \varepsilon \geq (1 - 2/N^2)(1-\eta)\phi_{\max}^t - \varepsilon. \tag{17}$$

Since there are finitely many pairs $(\hat{z}^t, \hat{\pi}^t)$, the algorithm will eventually repeat a pair provided it has not already terminated. Let τ be such an iteration, meaning $\hat{\pi}^\tau = \hat{\pi}^t$ and $\hat{z}^\tau = \hat{z}^t$ for some $t < \tau$. Hence

$$\begin{aligned}\Phi_L^\tau &\geq \sum_j \pi_j^t \phi_j(x^\tau|z^t) \geq \sum_j \hat{\pi}_j^t(\phi_j(x^\tau|\hat{z}^t) - \varepsilon) \geq \sum_j \hat{\pi}_j^t \phi_j(x^\tau|\hat{z}^t) - \varepsilon \\ &= \sum_j \hat{\pi}_j^\tau \phi_j(x^\tau|\hat{z}^\tau) - \varepsilon \geq (1 - 2/N^2)(1-\eta)\phi_{\max}^\tau - 2\varepsilon\end{aligned}$$

where the four inequalities follow, in order, from the cut added in iteration t, the definitions of $(\hat{z}^t, \hat{\pi}^t)$, the fact that the sum of the coordinates of $\hat{\pi}$ is at most 1, and the bound in (17).

Now suppose $\phi_{\max}^\tau \geq \lambda\phi_U^0$. Then $\dfrac{2\varepsilon}{\phi_{\max}^\tau} \leq \dfrac{2\varepsilon}{\lambda\phi_U^0} \leq \dfrac{\delta}{2}$ by our definition of ε, and in summary,

$$\frac{\phi_{\max}^\tau - \Phi_L^\tau}{\phi_{\max}^\tau} \leq \left[1 - (1 - 2/N^2)(1-\eta)\right] + \frac{2\varepsilon}{\phi_{\max}^\tau} \leq \frac{\delta}{2} + \frac{\delta}{2} = \delta \tag{18}$$

by choice of η. Thus (x^τ, Φ_L^τ) satisfies the relative convergence condition (b) on line 6 and the algorithm will terminate at iteration τ.

■

Corollary 2.9 *Suppose* $\Phi_{\min} \doteq \min_{x\in P}\Phi(x) > 0$. *Assuming uniform continuity of the* ϕ_i, *algorithm 1 terminates finitely.*

PROOF: Apply Theorem 2.8 using any $\lambda < \Phi_{\min}/\Phi_U^0$. ■

2.3.1 Choosing Θ

We first discuss a generic methodology for picking the Θ parameter. We will consider specific examples below. Let $x^* = \text{argmin}\{c(x) : x \in P\}$, i.e., an optimal solution to the nominal problem (1) (without loss of generality, $x^* = x^0$). Let $0 < \lambda^{\text{lo}} < \lambda^{\text{hi}} \leq 1$ and $0 < \xi$. Then we set

$$\Theta \;=\; \frac{c(x^*)\,\xi}{\Phi(x^*)\,(\lambda^{\text{hi}} - \lambda^{\text{lo}})}. \tag{19}$$

Lemma 2.10 *Suppose we run Algorithm 1 using (19). Assume that the algorithm does terminate in line 6 and let* $(\hat{x}, \hat{\Phi}_L)$ *be the iterate at termination.*

(i) If $c(\hat{x}) + \Theta\Phi(\hat{x}) \;\leq\; c(x^*) + \Theta\lambda^{\text{hi}}\Phi(x^*)$, *then*

$$\Phi(\hat{x}) \leq \lambda^{\text{hi}}\Phi(x^*) \quad \textit{and} \quad c(\hat{x}) \leq c(x^*)\left(1 + \frac{\lambda^{\text{hi}}}{\lambda^{\text{hi}} - \lambda^{\text{lo}}}\xi\right). \tag{20}$$

(ii) If $c(\hat{x}) + \Theta\hat{\Phi}_L \;>\; c(x^*) + \Theta\lambda^{\text{hi}}\Phi(x^*)$, *then*

$$\nexists x \in P \textit{ with } \Phi(x) \leq \lambda^{\text{lo}}\Phi(x^*) \;\textit{ and }\; c(x) \leq (1+\xi)c(x^*). \tag{21}$$

Proof. (i) Since $c(\hat{x}) \geq c(x^*)$ we obtain the first inequality in (20). Moreover, $\Phi(\hat{x}) \geq 0$ implies

$$c(\hat{x}) \;\leq\; c(x^*) + \lambda^{\text{hi}}\frac{c(x^*)\,\xi}{\lambda^{\text{hi}} - \lambda^{\text{lo}}} \;=\; c(x^*)\left(1 + \frac{\lambda^{\text{hi}}}{\lambda^{\text{hi}} - \lambda^{\text{lo}}}\xi\right). \tag{22}$$

(ii) If such an x existed, we would have $c(x) + \Theta\Phi(x) \leq (1+\xi)c(x^*) + \Theta\lambda^{\text{lo}}\Phi(x^*) \;=\; c(x^*) + \Theta\lambda^{\text{hi}}\Phi(x^*)$. This contradicts the fact that $\hat{x}$ is an optimal solution to the master problem in the last iteration of the algorithm.

■

Note that as per the above results (e.g., Lemma 2.2), the difference between $\hat{\Phi}_L$ and $\Phi(\hat{x})$ is small (and is near zero in the numerical examples below). In other words, the algorithm either generates a solution with both greatly reduced risk exposure and slightly increased cost, or proves that there is no point with moderately tighter requirements. As an additional point, exact solution of the master problem is only needed for part (ii).

Example. Let $\lambda^{\text{lo}} = 0.5$, $\lambda^{\text{hi}} = 0.6$ and $\xi = 0.01$. Under outcome (i) $\Phi(\hat{x}) \leq 0.6\,\Phi(x^*)$ while also $c(\hat{x}) \leq 1.06\,c(x^*)$. And under outcome (ii) there is no $x \in P$ with $\Phi(x) \leq 0.5\,\Phi(x^*)$ and $c(x) \leq 1.01\,c(x^*)$.

Discussion points.
(i) Lemma 2.10 can be used to develop a number of scaling or binary search algorithms that compute a vector with, both, risk exposure smaller than that of x^* by an approximate margin, and cost ramp-up that is likewise approximately constrained; or prove that no such vector exists.
(ii) The analysis in Lemma 2.10 assumes that the algorithm terminates in line 6. If not, by Theorem

2.8, after a finite number of iterations we compute a point $\tilde{x} \in P$ with $\Phi(\tilde{x}) \leq \lambda^{\text{lo}}\Phi(x^*)$. Should it be the case that $c(\tilde{x})$ is only slightly larger than $c(x^*)$ then (as in case (i) of the analysis directly above) we have computed a point with significantly lower risk exposure and slightly higher cost, as desired.

What if not, that is to say, what if $c(\tilde{x})$ is larger than $c(x^*)$ by a margin that is unacceptably high? In that case we appeal to Observation 2.4, i.e., there is *no* $x \in P$ with, both, $c(x) < c(\tilde{x})$ and $\Phi(x) \leq \Phi(\tilde{x})$. This is a similar situation as in case (ii) above – here we have a proof that the goal that we decrease risk exposure to as much as $\Phi(\tilde{x})$ requires a cost increase deemed excessive.

2.3.2 Note: the cases $\mathcal{Z} = \emptyset$ and $|\mathcal{Z}| = 1$

In either of these cases the optimization step in line 5 is void. Hence in line 8 we simply define $\pi_i^t = \frac{e^{\alpha\phi_i(x^t)}}{\sum_{j\in I} e^{\alpha\phi_j(x^t)}}$ for each $i \in I$.

2.3.3 Modifications to the algorithm

Here we describe a number of adaptations to Algorithm 1 that we have found effective from an empirical standpoint. In some cases, those modifications are also backed by theoretical guarantees.

Formally, a boosting kernel will be any algorithm that generates, at an iteration t, nonnegative *boosting weights* w_i^t for each $i \in I$, and a separation kernel on the other hand generates coefficients $\pi_i^t = \pi_i^t(w^t)$ with $\sum_{i\in I} \pi_i^t = 1$. A key point is that, for any separation kernel and any $\bar{z} \in \mathcal{Z}$, the inequality $\Phi_L \geq \sum_{i\in I} \pi_i^t \phi_i(x|\bar{z})$ is valid for the risk-adjusted problem (12).

Using this language, the boosting kernel in Algorithm (1) yields the boosting weights $\phi_i(x^t|z^t)$ where z^t is an approximate solution to the log-sum-exp maximization problem in line 5. The separation kernel applies the softmax function to the quantities $\phi_i(x^t|z^t)$. However, any combination of a boosting and a separation kernel yields a valid algorithm, and its effectiveness should be judged from an empirical perspective. We have experimented with several such variants, as follows.

Greedy method. Here the boosting kernel simply selects

$$(z^t, i^t) \;=\; \operatorname*{argmax}_{z\in\mathcal{Z}, i\in I} \phi_i(x^t|z)$$

and assigns weights $w_{i^t}^t = 1$ and $w_i^t = 0$ for all other i, and the separation kernel sets $\pi_{i^t}^t = 1$ and $\pi_i^t = 0$ otherwise, that is to say the cut added on line 9 is:

$$\Phi_L \;\geq\; \phi_{i^t}(x|z^t).$$

Note that $\pi^t = \text{SOFTMAX}(1, w^t)$.

Lemma 2.11 *Let $\pi \in (0,1)$. Consider Algorithm 1 using the greedy method, and suppose $0 < \lambda < 1$. Either this algorithm terminates finitely in line 6, or we reach an iteration t with $\phi_{\max}^t \leq \lambda\phi_U^0$.*

Comment. As compared with log-sum-exp, greedy boosting is simpler and can provide adequate performance. However log-sum-exp boosting is in general preferred (over greedy) because each resulting cut incorporates many features. Lemma 2.11 notwithstanding, we have observed, empirically, that log-sum-exp boosting yields faster termination, especially in high-dimensional settings where there are many ties or near-ties among features attaining high $\phi(x^t|z^t)$ value in line 5 at iteration t.

Flattening. The purpose of the boosting step at an iteration t is to highlight larger feature values in the resulting cut deployed in the separation step (i.e., larger $\phi_i(x^t|z^t)$ results in larger π_i^t). However, while the boosting accentuates order-of-magnitude differences in feature values, we also want the boosting to be less sensitive to small differences. Additionally, the numerical task in line 5 may become difficult when feature values are high. *Flattening*, when deployed, addresses both goals. It replaces each nominal feature value $\phi_i(x^t|z)$ by its logarithm $\ln \phi_i(x^t|z)$.

Synthetic boosting. When $\mathcal{Z} = \emptyset$, Algorithm 1 skips the boosting step altogether. An alternative is to rely on a *synthetic* set $Z^{\text{synth}} \subset \mathbb{R}^I_+$, and to define (for example) $\phi_i(x|z) = z_i + \phi_i(x)$. Hence the boosting step approximately solves

$$\max_{z \in \mathcal{Z}^{\text{synth}}} \sum_{i \in I} e^{\alpha(z_i + \phi_i(x^t))}. \tag{23}$$

Appropriate examples for $\mathcal{Z}^{\text{synth}}$ include "budgets" sets $\{z \in \mathbb{R}^I_+ : z_i \le \gamma_i \ \forall i, \sum_i z_i \le \Gamma\}$ and ball sets $\{z \in \mathbb{R}^I_+ : \sum_i z_i^2 \le \Gamma\}$.

Log-barrier term. When using synthetic boosting, we have found it empirically useful to add, to the objective function (23) log-barrier terms to prevent the optimal ζ_i values from reaching an extreme point of the set $\mathcal{Z}^{\text{synth}}$, i.e., to encourage the solution to the maximization problem to boost multiple features, which will as a result appear in the corresponding cut.

When, e.g., $\mathcal{Z}$ is the budgets set described above, the log-barrier contribution added to the boosting problem takes the form $\varepsilon \ln(\Gamma - \sum_i z_i) + \sum_i \varepsilon_i \ln(1 - z_i)$ for positive constants ε, ε_i ($i \in I$).

Clipping. Depending on the separation kernel, the cuts we obtain may be very dense when $|I|$ is large. This fact hinders the solution of the master problem. Additionally, many of the cut coefficient may be very small, further challenging solvers. *Clipping*, at an iteration t, works as follows:

1. Pick a (small) integer K.
2. Reset all but the top K values π_i^t to zero.
3. The top K π_i^t are proportionally reweighed so that they add to 1.

When $K = 1$, this amounts to using the cut $\Phi_L \ge \phi^t_{i^t}(x|z^t)$, where $i^t = \operatorname{argmax} \phi_i(x^t|z^t)$, which coincides with the greedy cut.

2.4 Introductory examples

Here we present simple examples where Algorithm 1 or heuristics are applied to optimization problems. More substantial computations are presented later.

2.4.1 Flow routing with queueing delays

Consider a minimum-cost flow problem with a single source s and a single destination t. There are m arcs, and each arc i has a capacity u_i and a per-unit flow cost c_i, and there are $M > 0$ units of flow to route from s to t.

In addition, when flow x_i is routed on arc i, the *queueing delay* experienced on this arc equals

$$\phi_i(x_i) = \mu(u_i, x_i) \doteq \frac{u_i}{u_i + \varepsilon - x_i},$$

where $\varepsilon > 0$ (and small) is used to avoid division by zero[3].

We apply Algorithm (1) using the cost function $c(x) \doteq \sum_i c_i x_i$ and impact function $\Phi(x) = \max_i \phi_i(x_i)$. (Thus, $\mathcal{Z} = \emptyset$.)

As an example, consider the network with two nodes (i.e., s and t) and three parallel arcs between s and t with data as follows

arc, i	1	2	3	4
capacity, u_i	70	31	50	50
per-unit cost, c_i	1	5	7	8

There are $M = 100$ units of flow to route. Thus, the nominal optimization to solve is:

$$\begin{aligned} \min\ & x_1 + 5x_2 + 7x_3 + 8x_4 \\ \text{s.t. } & x_1 + x_2 + x_3 + x_4 \ = \ 100 \\ & 0 \le x_1 \le 70,\ 0 \le x_2 \le 31,\ 0 \le x_3 \le 50,\ 0 \le x_4 \le 50, \end{aligned}$$

with optimal solution $x^0 = (70, 30, 0, 0)$ attaining cost 220. We set $\varepsilon = 0.01$. Thus, the maximum queueing delay is attained on arc 1 and it equals $\mu(70, 70) = 100$.

To obtain the master problem we need to add the Φ_L variable to the nominal optimization problem; additionally, we need to represent the quantity $\phi_i(x_i)$ for each $i = 1, 2, 3$. This is done by introducing a variable q_i as a proxy for $\phi_i(x_i)$ together with the inequalities

$$q_i \ \ge \ \mu(u_i, x_i) \quad (= u_i/(u_i + \varepsilon - x_i)), \quad i = 1, 2, 3,$$

which are convex-representable.

Following section 2.3.1 to set Θ, we choose $\lambda^L = 0.5$, $\lambda^U = 0.6$ and $\xi = 0.01$. Using $c(x^0) = 220$ and $\Phi(x^0) = 100$, we obtain $\Theta = 0.22$. In summary, the master problem is initialized as:

$$\begin{aligned} \min\ & x_1 + 5x_2 + 7x_3 + 8x_4 \ + \ 0.22\,\Phi_L \\ \text{s.t. } & x_1 + x_2 + x_3 + x_4 \ = \ 100 \\ & q_1 \ge \mu(70, x_1),\ q_2 \ge \mu(30, x_2),\ q_3 \ge \mu(50, x_3),\ q_4 \ge \mu(50, x_4) \\ & 0 \le x_1 \le 70,\ 0 \le x_2 \le 31,\ 0 \le x_3 \le 50,\ 0 \le x_4 \le 50,\ \Phi_L \ge 0. \end{aligned}$$

Finally, we heuristically set $\alpha = 1/10$. The algorithm proceeds as follows.

1) Since $\mathcal{Z} = \emptyset$, line 5 of the algorithm simply amounts to evaluating $\Phi(x^0)$. As per line 8 of the algorithm, the cut we obtain on line 9 is (values rounded to three digits)

$$\Phi_L \ge 0.980\, q_1 + 0.018\, q_2,$$

where we have ignored very small (but positive) coefficients on x_3 and x_4.

2) Resolving the master problem with the added cut yields the solution vector $x_1^1 = 69.005$, $x_2^1 = 30.995$, $x_3^1 = x_4^1 = 0$ and $\phi_L^1 = 41.266$; its cost is $c(x^1) = 223.980$. Evaluating queueing delays, we have $\Phi(x^1) = 41.297$.

We note the small gap between ϕ_L^1 and $\Phi(x^1)$: the formal algorithm is close to termination as per line 6. However, the central point here is that

[3]The function $\mu(u, x)$ is the M/M/1 queuing delay on a channel with capacity u and data rate x. See [25]

(a) Comparing $\Phi(x^0) = 100$ and $\Phi(x^1) < 41.30$ we see greatly reduced risk.

(b) Comparing $c(x^0) = 220$ and $c(x^1) < 223.99$, we see that cost has increased by less than 1.5%.

As per our goal of "substantially reducing risk without materially increasing cost" we can terminate the algorithm. Note that both (20) and (22) are satisfied.

2.4.2 Min-cost flow under capacity reduction

Consider the generic single-source, single-destination min-cost flow setup with M units of flow to route. Here we consider the case where there is a set $\mathcal{Z}$ of *decreased capacity vectors*. Each $z \in \mathcal{Z}$ has an entry z_{uv} corresponding to each arc (u,v), with value indicating the capacity of (u,v) in an adverse scenario that is realized after the choice of flow vector x has been made, and which impacts multiple arcs at once.

Specifically, given a flow vector x, we assume that the capacity in each arc (u,v) is (adversarially) reduced to $\min\{x_{uv}, z_{uv}\}$. As a result, some of the M units flow originally routed from s to t using flow vector x may be lost, even if we reroute using the reduced capacities. This model incorporates a form of recourse which we have selected for this example to highlight the behavior of our algorithm. An alternative would have been to use capacities z_{uv} rather than $\min\{x_{uv}, z_{uv}\}$ – the version we selected is more constrained in terms of recourse.

Formally, given a flow vector x we proceed as follows. For each vector $z \in \mathcal{Z}$,

(a) Let $F(x|z)$ denote the maximum flow that can be routed from s to t in the network where each arc (u,v) has *effective capacity* $\min\{x_{uv}, z_{uv}\}$. Then $F(x|z) \le M$.

(b) For each subset of the nodes S with $s \in S$ and $t \notin S$ let $\delta^+(S)$ be the set of arcs (u,v) with $u \in S$ and $v \notin S$. We write $\text{cap}_S(x|z) = \sum_{(u,v)\in\delta^+(S)} \min\{x_{uv}, z_{uv}\}$ to denote the *effective capacity* of the cut, i.e., the cut-capacity using the arc capacities in (a). For such a cut define $\phi_S(x|z) = \max\{M - \text{cap}_S(x|z), 0\}$. In our terminology, there is a feature for each cut.

(c) Let $\mathcal{C}$ be the set of cuts as in (b). We set $\phi(x|z) = \max_{S\in\mathcal{C}} \phi_S(x|z)$. We have $\phi(x|z) > 0$ if the effective capacities $\min\{x_{uv}, z_{uv}\}$ do not allow M units of flow to be routed from s to t.

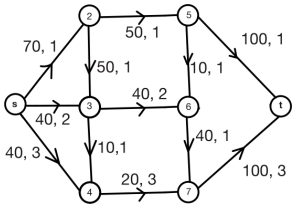

Figure 3: Reduced capacity example. For each arc we list capacity and per unit cost

In summary, we write $\Phi(x) = \max_{z\in\mathcal{Z}} \phi(x|z)$ where we assume that $\mathcal{Z}$ is compact[4]. If $\mathcal{Z}$ is finite the risk-adjusted problem (7) has an explicit linear representation. However, if $\mathcal{Z}$ is large this representation will be prohibitively large both in terms of constraints and variables.

Next we describe our implementation of these ideas in a numerical example, given by the network in Figure 3, where we have to route $M = 100$ units of cost from s to t.

For the sake of simplicity, the reduced capacity model we use is that in each $z \in \mathcal{Z}$ up to two arcs of the network have their capacity reduced by (exactly) 50% with no other reductions. For a flow vector x the computation of $\Phi(x)$ can be modeled by a linear mixed-integer program, using binary variables to indicate the arcs where capacity is reduced, and using the dual representation of the min-cut problem – the *dual MIP* in what follows.

[4]If $\mathcal{Z}$ is finite and each $z \in \mathcal{Z}$ has a probability p_z then we could instead consider $\Phi(x) = \sum_{z\in\mathcal{Z}} p_z \phi(x|z)$. An example is that where we allow a full min-cost flow recomputation once the decreased capacities are realized.

We use the greedy method (Section 2.3.3) and we add only one inequality to the master at each iteration. After initializing the master problem as the min cost flow problem on the given network the algorithm proceeds as follows:

Iteration 0. Nominal solution x^0.

arc	$(s,2)$	$(s,3)$	$(s,4)$	$(2,3)$	$(3,4)$	$(2,5)$	$(3,6)$	$(4,7)$	$(5,6)$	$(6,7)$	$(5,t)$	$(7,t)$
flow	60	40	0	10	10	50	40	10	0	40	58	50

The optimal cost is $c(x^0) = 560$. The dual MIP sets $z^0_{s,2} = 35$ and $z^0_{s,3} = 20$ as the two reduced capacities. The cut separating s from the rest of the network is a min effective capacity cut, with effective capacity $z^0_{s,2} + z^0_{s,3} + x_{s,4} = 55 + x^0_{s,4} = 55$. Hence $\Phi(x^0) = 100 - 55 = 45$. Hence we add, to the master problem, the cut $\Phi_L + 55 + x_{s,4} \geq 100$.

Also, using $\lambda^{\text{lo}} = 0.7$, $\lambda^{\text{hi}} = 0.75$ and $\xi = 0.01$ (19) sets $\Theta = 2.49$.

Iteration 1 yielding x^1.

arc	$(s,2)$	$(s,3)$	$(s,4)$	$(2,3)$	$(3,4)$	$(2,5)$	$(3,6)$	$(4,7)$	$(5,6)$	$(6,7)$	$(5,t)$	$(7,t)$
flow	70	10	20	20	0	50	30	20	0	30	50	50

This solution has cost 570 and it estimates $\Phi_L = 25$. The dual MIP sets $z^1_{s,2} = 35$ and $z^1_{2,5} = 25$, and the minimum effective-capacity cut separates node s from the rest of the network. Its effective capacity equals $z^1_{s,2} + x^1_{s,3} + x^1_{s,4} = 35 + 30 = 65$ and therefore $\Phi(x^1) = 35$. We add to the master the inequality $\Phi_L + x_{s,3} + x_{s,4} \geq 65$.

Iteration 2 yielding x^2.

arc	$(s,2)$	$(s,3)$	$(s,4)$	$(2,3)$	$(3,4)$	$(2,5)$	$(3,6)$	$(4,7)$	$(5,6)$	$(6,7)$	$(5,t)$	$(7,t)$
flow	60	20	20	10	0	50	30	20	0	30	50	50

The solution to the master problem has cost 570 and $\Phi_L = 25$. The min-effective capacity cut now separates nodes $s, 2, 3,$ and 4 from the rest, with $z^2_{2,5} = 25$ and $z^2_{3,6} = 20$, and effective capacity $45 + x^2_{4,7} = 20 = 65$. Hence $\Phi(x^2) = 35$. Note that the reduced capacity of the adversarial cut is $25 + 20 + x_{4,7} = 45 + x_{4,7}$; however, $x^2_{3,6} = 30$, so x^2 only loses 10 units of flow when the capacity of $(3,6)$ is reduced to 20 units (as in the current dual MIP solution). The inequality we add to the master is $\Phi_L + x_{4,7} \geq 45$.

Iteration 3 yielding x^3.

arc	$(s,2)$	$(s,3)$	$(s,4)$	$(2,3)$	$(3,4)$	$(2,5)$	$(3,6)$	$(4,7)$	$(5,6)$	$(6,7)$	$(5,t)$	$(7,t)$
flow	70	20	10	20	10	50	30	20	0	30	50	50

The solution to the master has cost $c(x^3) = 570$ and $\Phi_L = 35$. The dual MIP now sets $z^3_{s,2} = 35$ and $z^3_{2,5} = 25$, but only achieves $\Phi(x^3) = 35$ units of lost flow (again using the cut that separates s from the rest of the network). Since $\Phi(x^3) = \Phi_L$ the algorithm terminates.

In summary,

$$\Phi(x^3)/\Phi(x^0) = 35/45 \approx 0.78 \text{ and } c(x^3)/c(x^0) = 570/560 \approx 1.0179.$$

At the same time, we note that $c(x^3) + \Theta\Phi(x^3) = 570 + 2.49 * 35 = 657.15 > 644.0375 = c(x^0) + \Theta\lambda^{\text{hi}}\Phi(x^0)$. Consequently, by Lemma 2.10 (ii), we conclude that

$$\nexists \text{ feasible flow } x \text{ with } \Phi(x) \leq 0.7\,\Phi(x^0) \text{ and } c(x) \leq 1.01\,c(x^0).$$

3 Numerical experiments

Here we describe testing of our algorithms on large problem instances arising in two settings: operation of power grids, and logistics.

3.1 Logistics under concentration risk

In this section we apply our methodology to an evolved version of the logistical setup described in [24], which arose from an engagement with a stakeholder, and where a heuristic version of Algorithm 1 was applied. In brief, the problem being considered has the following attributes.

- Multiple commodities are shipped from sources to destinations in a network, using multiple vehicle types. The different commodities have numerical priorities and delivery time windows with lateness penalties. Vehicles have restricted availability (when and where).
- In the numerical instance considered here, there are 44 time periods; the network has 17 nodes and 224 links; there are 94 commodity types and 6 vehicle types.
- The resulting MIP formulation has 417,330 variables of which 5,920 are integral and there are 118,343 constraints. Gurobi v12 [21] was used to obtain a solution within 5% of optimality while requiring a few minutes of computation.
- It was observed, across multiple instances of the problem, that the computed MIP solution exhibits a high degree of concentration; see Figures 1 and 2. This feature was deemed (by the stakeholder) to be risk-inducing. In a humanitarian logistics setting, or any setting addressing the delivery of critical items, concentration of resources at a geographical location during a short time span is viewed as risky because the loss of a significant quantity of valuable assets due to unforeseen exogenous events (no matter their nature) is of high impact.

We have applied the methodology in this paper as follows. First, for a solution vector x in the MIP, and for an arc (i,j) and time period τ, we write $\phi_{i,j,\tau} = \phi_{i,j,\tau}(x)$ total weight of all commodities shipped on (i,j) starting at time τ under solution x. Here, a commodity's "weight" is defined as the product of the numerical priority times the actual shipped weight. We have also considered, using the same methodology, the case where $\phi_{i,j,\tau}$ only accounts for shipped weight.

For these experiments we used the following version of Algorithm 1:

1. The risk term we focused on was $\Phi(x) = \max_{i,j,\tau} \phi_{i,j,\tau}(x)$, i.e., max shipped weight, on any link and any point in time.
2. We used flattening, synthetic boosting, and log-barrier terms – see Section 2.3.3 and equation (23). Flattening is needed because the $\phi_{i,j,\tau}$ values can be large, as can be the number of triples (i,j,τ), and we used a synthetic set because there is no overt uncertainty in the model. We relied on $\mathcal{Z}^{\text{synth}} = \{\zeta_{i,j,\tau} \geq 0 : \sum_{i,j,\tau} \zeta_{i,j,\tau} \leq 1\}$. Furthermore, we used $\alpha = 1$. Thus, at iteration t of Algorithm 1, the boosting step (approximately) solves:
$$\max_{\zeta \in \mathcal{Z}^{\text{synth}}} \left[\sum_{i,j,t} e^{\zeta_{i,j,\tau}} \phi_{i,j,\tau}(x^t) \quad + \quad \varepsilon \ln\left(1 - \sum_{i,j,\tau} \zeta_{i,j,\tau}\right) + \sum_{i,j,\tau} \varepsilon_{i,j,\tau} \ln(1 - \zeta_{i,j,\tau}) \right], \tag{26}$$
where $\varepsilon > 0$ and $\varepsilon_{i,j,\tau} > 0$ for every triple (i,j,τ).
3. This optimization task was addressed using a first-order algorithm detailed below in Section 3.1.1. The algorithm was implemented in multistart fashion, using parallel processing.

4. Each start yields a different vector $\zeta \in \mathcal{Z}^{\text{synth}}$. The resulting SOFTMAX cut was added to the master problem if violated.

5. We used $\Theta = c(x^*)/\Phi(x^*)$. This is consistent with (19) when $\xi = \lambda^{\text{hi}} - \lambda^{\text{lo}}$.

6. Algorithm 1 was run using the LP-relaxation of the MIP, solved using the Barrier method (without crossover) in Gurobi [21]. Upon termination, the MIP, with cuts added, was then run.

Iteration	Risk	Cost	Cuts	runtime (s)
0	66724	1095	132	25.62
1	53992	1179	196	28.84
2	39969	1153	216	30.59
3	50437	1153	229	30.61
4	50385	1153	234	30.77
5	44181	1156	234	29.13
MIP	44181	1186		91.65

We highlight that the MIP attains a 33.33% decrease in risk metric while incurring an 8.3% increase in cost. In fact, in hindsight, we could have terminated the algorithm in iteration 2 where the performance is clearly better both in terms of risk and cost.

Table 1: Comparison: cumulative weight shipped in top activities, in units of 10^5

# of activities, K	1	2	4	8	10	16	32	64	100
nominal	3.45	6.79	12.19	20.51	23.99	33.24	45.08	52.72	53.72
risk-aware	1.45	2.91	5.82	11.63	14.54	23.26	38.37	50.80	54.23

Table 1 complements Figure 2. It displays the weight shipped in the top K activities for both the nominal and risk-aware solutions, for selected values of K up to 100, which accounts for nearly all the shipments. We remind the reader that some 150 activities have positive weight in both solutions (out of, potentially, over $10,000$). We highlight the high concentration evident in the nominal solution: for example, the top ten activities account for nearly 50% of all shipped weight. In contrast, the risk-aware solution reduces the concentration metric by roughly 50 for $K < 10$, and gradually catches up with the nominal solution – while incurring a relatively small increase in cost.

3.1.1 First-order method for the boosting step

The boosting problem (26) is solved via a custom multi-start first-order ascent algorithm (henceforth termed the *Red* solver) based on AdaDelta [35], so as to handle the optimization problem (26). At the time of this writing, the existing optimization algorithms in Knitro [13], a local nonlinear solver, exhibit difficulty converging on our instances due to the highly nonlinear interaction between the exponential and log-barrier terms; our custom approach exhibits superior performance. A custom multi-start, first-order algorithm based on AdaDelta proves superior to other popular first-order algorithms for its ability to self-calibrate the sensitive choice of step size, and because it does not use momentum which often pushes iterates past the log-barrier. Multi-starting generates multiple runs per boosting step. We provide further details next.

The feature values, $\phi_{i,j,\tau}(x^t)$, of the current solution are normalized and scaled by a predetermined constant to prevent either the boosting term or log-barrier terms from dominating the objective. We mutli-start the first-order solver with a number of initial points, n_{starts}. For each of these

starts $1 \leq h \leq n_{\text{starts}}$ we select an initial point $\zeta^{(h)}$ using a sliding window, $\mathcal{W}_h$, of given size and step, on the sorted indices of the feature values $\phi_{i,j,\tau}(x^t)$ via,

$$\zeta^{(h)}_{i,j,\tau} = \begin{cases} w & (i,j,\tau) \in \mathcal{W}_h \\ 0 & (i,j,\tau) \notin \mathcal{W}_h \end{cases},$$

where $w > 0$ is some small constant.

The motivation for using a sliding window is twofold. Firstly, in high dimensions, the sum $\sum_{i,j,\tau} \zeta^{(h)}_{i,j,\tau}$ easily surpasses the the budget of 1, i.e., violates the aggregate log-barrier, even for near-zero efforts across each feature. So we only initialize a subset of variables as nonzero; namely, those corresponding to the largest feature values. Secondly, the largest value $\zeta^{(h)}_{i,j,\tau}$ in a solution, likely corresponds to the largest feature value $\phi_{i,j,\tau}(x^t)$, which, if only the cut associated with $\zeta^{(h)}$ is added, can cause the next solution, x^{t+1}, to shift weight away from the tuple (i,j,τ). The sliding window masks larger features from previous starts in order to elucidate the next greatest exposure to concentration in a look-ahead-type fashion.

Optionally, indices of $\zeta^{(h)}$ corresponding to the largest feature values within the window are set to a larger constant $w' > w$. This mitigates the solver converging to local optima where equal weight is put on multiple features, when an asymmetric solution focusing on the features with higher concentration is superior. Moreover, during each start h, the gradients of coordinates $\zeta^{(h)}_{i,j,\tau}$ outside of the window are filtered out of the first-order update computation, thus fixing these coordinates to 0 during the course of the solve.

Worker processes are used to parallelize the solve of each initial point via a custom implementation of AdaDelta using projection and backtracking. Projection is used to prevent variables from becoming negative, and is implemented by clipping, i.e., taking the positive part of the iterate $[\zeta^{(h)}]^+$, after each gradient step. Conversely, backtracking is employed if a variable violates the log-barrier, and is implemented so as to satisfy Wolfe conditions [29].

The disparity in which corrective measure to use for maintaining feasibility is based on our belief that ζ variables approaching the log-barriers indicate features worth disrupting and so merit a delicate analysis, whereas those approaching 0 are unappealing to the solver and can be discarded.

We now specify the parameters and settings for the Red solver used in our experiments. The log-barrier constants are $\varepsilon = \varepsilon_{i,j,\tau} = 1.0$, for the simple reason that the log-barriers tended to dominate the objective for larger values. With this choice of parameters, normalizing and scaling the features of the current solution, $\phi_{i,j,\tau}(x^t)$, by 10.0 provides a balanced interaction between the boosting and log-barrier terms in the objective.

For multistarting we used $h^{\text{starts}} = 30$, a window of size 10 and step 1, and value $w = 1/(10 \dim W)$. A window size of 10 enhances the first-order solver focus on the highest concentration for each start. The constant $\frac{1}{10 \dim W}$ induces the initial sum of ζ variables, $\frac{1}{\dim W}$, to be much less than the aggregate budget 1. Moreover, we warmstarted 1 coordinate (the largest feature in the window) to the value $w' = 0.5$. Finally, the internal parameters for AdaDelta and backtracking are the default values in their respective literature [35, 29].

3.2 Constraint coefficient uncertainty

Now consider problems of the form (4) where the constraint coefficients, a_{ij}, are subject to uncertainty. Such uncertainties can arise in problems whose formulations rely on physical measurements, such as in power flow [14], pooling [27], and beamforming [34]. See [33] for a more extensive analysis.

For experimentation, we consider instance QPLIB 2894 of the Quadratic Programming Library [19], which is a linear quadratically constrained program with 17 variables and 209 constraints. Many constraints, both linear and quadratic, feature fractional coefficients that could contain errors. For example, see constraint e198 (with coefficients rounded to nearest thousandth) below.

$$
\begin{aligned}
&22.009x_6^2 + 9.132x_6x_8 + 18.458x_6x_9 - 11.482x_6x_{10} - 9.131x_6x_{11} - 2.156x_6x_{16} + 2.156x_6x_{17} \\
&+ 22.009x_7^2 + 2.156x_7x_8 + 11.482x_7x_9 + 18.458x_7x_{10} - 2.156x_7x_{11} + 9.132x_7x_{16} - 9.132x_7x_{17} + x_8^2 \\
&+ 4.391x_8x_9 - 1.478x_8x_{10} - 2x_8x_{11} + 5.367x_9^2 - 4.391x_9x_{11} + 1.478x_9x_{16} - 1.478x_9x_{17} + 5.367x_{10}^2 \\
&+ 1.478x_{10}x_{11} + 4.391x_{10}x_{16} - 4.391x_{10}x_{17} + x_{11}^2 + x_{16}^2 - 2x_{16}x_{17} + x_{17}^2 \;\geq\; 1.300
\end{aligned}
$$

At the optimal solution x^*, $x_7^* = 1.000$ and $x_{17}^* = 8.450$, so the term $-9.132x_7^*x_{17}^* = -77.165$. Moreover, this constraint has a slack of 0.024. Should the coefficient -9.132 of the monomial x_7x_{17} decrease by 1% to -9.223, the value of the monomial would become -77.934, and x^* would violate the constraint by 0.745 or 57.5% of the right-hand side. In fact, any error in this coefficient beyond 0.04% in the negative direction results in a violation.

To measure the adversarial infeasibility incurred by x feasible for (4), we use

$$
\phi_i(x|z) \;=\; \frac{1}{s_i}\left[\sum_{j\in J_i} a_{ij}f_{ij}(x)(1+z_{ij}) - b_i\right]^+ \;\doteq\; \frac{1}{s_i}\left[\sum_{j\in J_i} a_{ij}f_{ij}(x)z_{ij} - \text{slack}_i(x)\right]^+,
$$

where $z_{ij} \geq 0$ is the error on coefficient a_{ij}, $s_i \geq 0$ is some scale factor for constraint i, and $\text{slack}_i(x) = b_i - \sum_{j\in J_i} a_{ij}f_{ij}(x)$ is the slack of constraint i for solution x. We examine the two scaling choices: $s_i = \|a_i\|_\infty$ for "scaled absolute violation" and $s_i = \max(b_i, 1)$ for "percent violation". For the adversarial budget we used the Euclidean ball $\mathcal{Z} = \{z \in \mathbb{R}^d : \|z\|_2 \leq .01\}$, where $d = |I|\sum_{i\in I}|J_i|$, hence limiting individual coefficient error to within 1% in the worst case. Finally, we used the following implementation of Algorithm 1:

- We compute parameter Θ as per (19), with $\xi = 0.01$, $\lambda^{\text{lo}} = 0.48$ and $\lambda^{\text{hi}} = 0.50$.
- The boosting step used the greedy method, i.e., at iteration t we set

$$
z^t \;=\; \operatorname*{argmax}_{z\in\mathcal{Z}:\,\|z\|_0=1} \max_{i\in I} \frac{1}{s_i}\left[\sum_{j\in J_i} a_{ij}f_{ij}(x)z_{ij} - \text{slack}_i(x)\right]^+,
$$

  which is the unit vector $.01e_{i^t j^t}$ for the coordinate $(i^t, j^t) \in I \times J_i$ that maximizes $\frac{1}{s_i}\left[.01(a_{ij}f_{ij}(x)) - \text{slack}_i(x)\right]^+$.
- For the cutting step we add the cut $\Phi_L \geq \phi_{i^t}(x|z^t) = \left[.01(a_{i^t j^t}f_{i^t j^t}(x)) - \text{slack}_{i^t}(x)\right]^+ / s_{i^t}$.
- The algorithm is run until it terminates as per criterion b on line 6.
- The master problem is solved using Gurobi [21].

We present results for both measures of adversarial infeasibility in tables 2 and 3. Revisiting the monomial x_7x_{17} in constraint e198, the potential 57.5% violation is the initial worst-case for the percent violation measure of infeasibility. Upon termination of the algorithm, the new optimal $\hat{x}$ has $\hat{x}_7 = 0.998$ and $\hat{x}_{17} = 8.472$, so the term $-9.132\hat{x}_7\hat{x}_{17} = -77.212$. However, this constraint now has a slack of 0.717, so a 1% decrease in the coefficient -9.132 only results in violation of 4.03% of the right-hand side. Moreover, any error in this coefficient smaller than 0.92% will not result in a violation of constraint e198 for $\hat{x}$.

Preliminary version: Wed.June.30.165033.2026

Table 2: Results for scaled absolute violation

iterations	runtime (s)	worst-case scaled abs. violation	corresp. percent violation, %	cost Δ, %
0	2.76	0.069	4.80	-
2	11.96	0.034	12.20	1.00
7	32.65	0.014	4.20	1.06

Table 3: Results for percent violation

iterations	runtime (s)	worst-case percent violation ,%	corresp. scaled abs. violation	cost Δ, %
0	2.78	57.53	0.034	-
1	7.38	17.85	0.045	1.07
6	19.67	4.23	0.002	1.08

3.3 ACOPF under large branch loading and thermal risk

The ACOPF (Alternating Current Optimal Power Flow) problem arises in the operation of power grids, and has received considerable attention in part due to the increasing importance of efficient energy delivery, and also because of a perceived increase in failures of grids. To put the risk issue into perspective, a standard operating requirement for power grids is that they be able to tolerate the failure of any one component (for any reason whatsoever) and this requirement is incorporated into optimization algorithms in daily use; the so-called N-1 rule.

A more evolved risk attitude focuses on a variety of flavors, including thermal management issues, and we address that perspective in this section. A salient fact is that imposing a strict "N-K" requirement for $K > 1$ but small (e.g., $K = 3$) would be onerous, and an alternative is to reduce rather than eliminate risk, if it can be done at small cost.

For background on ACOPF and related issues, see [15], [14], [7], [8], [28], [16]. For the purpose of explaining the experiments described here, we provide a brief outline of the problem and its risk implications, using a mix of power systems and generic terminology.

- The problem seeks to generate power at minimum cost, subject to satisfying a set of demands, and physical constraints.
- As input to the problem there is a network (in the graph theoretic sense) whose *branches* (i.e., arcs) model transmission lines and are endowed with numerical parameters that describe the power flow physics.
- Power is generated at the nodes of the network, where generators are housed; each generator is described by a (convex) cost function and a maximum output. Demands are also situated at the nodes of the network.
- Power flows on branches are described by nonlinear and nonconvex equations involving physical quantities (voltages) which are also variables and are nodal attributes.
- For each branch (k,m) the power flowing from k to m is indicated by a variable P_{km}[5].
- Each branch (k,m) has a *resistance* $r_{km} > 0$, and the thermal energy radiated by the line is, approximately, $r_{km} P_{km}^2$ under standard assumptions on voltages (a more accurate representation is provided by the *current* on branch (k,m)).

[5] A technical detail is that, in general, $P_{km} \neq -P_{mk}$ though the difference is small.

- There is a standard approximation (not a relaxation) to ACOPF that is linear: the so-called DC approximation, or DCOPF in short, which is in fact the standard formulation used in economic grid operation (even if ACOPF is more accurate and used primarily for generic risk assessments). DCOPF does include quantities playing the role of the P_{km} above, and has identical cost function as ACOPF.

In the experiments that we describe in this section, cost (i.e., the $c(x)$ in our formulations) will be the generation cost. We will use x to generically refer to the set of all variables in ACOPF. To describe the impact function $\Phi(x)$ we rely on setups using the $r_{km}P_{km}^2$ thermal quantities described above. In what follows we write

$$T_{km} = T_{km}(x) \doteq r_{km}P_{km}^2$$

for brevity. Also let $\mathcal{B}$ denote the set of branches of the network. Here we describe four different experiments, using a combination of $\Phi(x)$ functions, without and with a set $\mathcal{Z}$, and using different boosting and separation approaches. Some common details are as follows:

1. In all cases we compute the quantity Θ as per (19): $\Theta = \frac{c(x^*)\,\xi}{\Phi(x^*)\,(\lambda^{\text{hi}}-\lambda^{\text{lo}})}$, with $\xi = 0.01$, $\lambda^{\text{lo}} = 0.4$ and $\lambda^{\text{hi}} = 0.5$.

2. To set α in Algorithm 1 we used the following heuristic (which is aligned with the theory presented above). Let $T^* = \max_{km} T_{km}$ in the nominal case. Then $\alpha = \min\{50, \ln|\mathcal{B}|/(0.25\Phi(x^*))\}$. The denominator in this last expression was motivated by empirical observations using relevant examples.

3. The algorithm is run until it terminates, or it achieves $\Phi_L \leq 0.5\,\Phi(x^0)$.

4. The algorithm is run using the DCOPF formulation which is addressed using Gurobi. Upon termination, the cuts are transferred to the ACOPF formulation and we run Knitro [13] using the amplpy [1] interface to handle that problem. We also run Knitro on the nominal ACOPF formulation in order to obtain the nominal cost and risk values.

5. With regards to the item above, an important perspective here is that while the AC formulation is clearly better aligned with power flow physics, the energy markets actually rely on the DC formulation. In light of that fact the DC-based approach is in a precise sense more meaningful, but we include the AC-based analysis for completeness.

In the experiments described below, we used the following well-known cases present in the Matpower [36] or pglib [2] libraries:

#	name	nodes	branches	generators
1	pglib_opf_case13659_pegase__api.m	13659	20467	4092
2	pglib_opf_case30000_goc__api.m	30000	35393	3526

These cases have the following attributes:

#	variables, AC	constraints, AC	variables, DC	constraints, DC
1	137837	170587	51878	47785
2	222878	307751	98920	95393

3.3.1 Introductory case with $\mathcal{Z} = \emptyset$ and maximum thermal metric

First we use

$$\Phi(x) \;=\; \max_{(k,m)\in\mathcal{B}} T_{km}.$$

Thus, there is a feature per branch, and since $\mathcal{Z} = \emptyset$ the boosting step (line 5 of Algorithm 1) is void. In terms of the cut used in the algorithms, we define

$$\pi_{km} \;=\; \frac{e^{\alpha T_{km}}}{\sum_{(k',m')\in\mathcal{B}} e^{\alpha T_{k'm'}}},$$

that is to say, $\pi = \text{SOFTMAX}(1, T)$. The coefficients π_{km} are rounded to zero if they fall beneath 10^{-6}. We obtained the following results; in all tables below "cost Δ" stands for cost increment.

#	iterations	runtime (s)	risk reduction, %	cost Δ, %	AC risk reduction, %	AC cost Δ, %
1	4	5.63	50.18	0.56	31.91	0.44
2	3	6.6	47.86	0.80	28.64	0.48

Note the relative degradation of the improvement in the risk metric when we go from the DC to the AC formulation.

3.3.2 Max thermal metric under distributionally robust exogenous risk

Next we consider a variant of the above max thermal feature with a significant difference. Namely, we assume that the thermal metric $T_{km} = r_{km}P_{km}^2$ is now augmented with a term that reflects an exogenous source of risk. Briefly, elevated transmission line temperatures pose a number of risks that are difficult to quantify. Additionally, line temperatures are due to a combination of endogenous factors (our T_{km} term, in approximation) and exogenous and uncertain factors that are extremely difficult to precisely specify and measure. See the IEEE 738 standard [22].

Both factors, endogenous and exogenous, are nominally combined using the heat equation, whose (deterministic) solution would yield line temperature as a function of time – once more, an effectively impossible task due to data unavailability to the operator. As a final point, it is worth stressing that under such a model, actual line temperature on an branch (k,m) is a highly nonlinear function of T_{km} and the exogenous factors.

We take a stance where we measure actual risk by adding, to the T_{km} term, a second term that reflects a given multiple of a standard deviation of additional thermal risk exposure. We assume that to first order, that additional term is *proportional* to T_{km}.

More precisely, for a branch (k,m), we write $\phi_{km}(x|z) = (1 + z_{km})T_{km}$, where $z_{k,m} \geq 0$ is the multiplicative exogenous impact on risk. In this section we focus on the case where

$$\Phi(x) \;=\; \max_{z\in\mathcal{Z}} \max_{(k,m)\in\mathcal{B}} \phi_{km}(x|z),$$

and we will use a budgets set

$$\mathcal{Z} \;=\; \left\{ 0 \leq z_{k,m} \leq 1 \,:\, \sum_{(k,m)\in\mathcal{B}} z_{k,m} \leq N \right\},$$

for some $N > 0$. In terms of the usage of Algorithm 1, we applied the following rules:

- The solution to the boosting problem in line 5 is obtained by setting $z_{km} = 1$ for the N branches (k,m) with top value $T_{k,m}$, and $z_{k,m} = 0$ otherwise.

- We used $N = 5$, and we applied *clipping* (Section 2.3.3) when computing the cut used in the separation step. This last feature is motivated by the fact that at any given iteration t, the branches with $z^t_{k,m} = 1$ are also those with largest value $T_{k,m}$, and, as a result, the sum of the top $2N$ values $e^{\alpha(1+z_{km})T_{km}}$ strongly dominates the sum of all such terms. In other words, the top $2N$ values π^t_{km} already add up to approximately 1 (and the rest contribute a negligible amount).

#	iterations	runtime (s)	risk reduction, %	cost Δ, %	AC risk reduction, %	AC cost Δ, %
1	10	18.78	50.76	1.05	36.41	0.56
2	5	64.66	50.33	0.56	28.43	0.48

3.3.3 Average of top ten

Let $T_{(i)} = T_{(i)}(x)$ be the i^{th} largest value T_{km}. We remind the reader that the $T_{km} = r_{km}P^2_{km}$ and that x is the vector of all variables, including the P_{km}. In the next set of experiments we use

$$\Phi(x) \;=\; \frac{1}{10}\sum_{i=1}^{10} T_{(i)}.$$

This is again a case with $\mathcal{Z} = \emptyset$. If we use $\mathcal{B}^{10}$ to denote the set of all 10-tuples chosen from $\mathcal{B}$, we can equivalently write

$$\Phi(x) \;=\; \max_{s\in\mathcal{B}^{10}} \phi_s(x), \quad \text{where for } s \in \mathcal{B}^{10} \text{ we write } \phi_s(x) = \frac{1}{10}\sum_{(k,m)\in s} T_{km}. \tag{27}$$

In other words, there is a feature for each 10-tuple of branches. For the separation step we used the following approach:

- At any iteration t we generate a set $S^t \subset \mathcal{B}^{10}$.
- We construct S^t by selecting 10-tuples chosen from among the top 20 branches (k,m) as per the metric T_{km}. The set S^t is guaranteed to include the tuple s attaining maximum value $\phi_s(x^t)$ as in (27).
- For each tuple $s \in S^t$ we add the cut $\Phi_L \geq \frac{1}{10}\sum_{(k,m)\in s} T_{km}$. Thus, with $|S^t| = 10$, ten cuts are added in each iteration.

#	iterations	runtime (s)	risk reduction, %	cost Δ, %	AC risk reduction, %	AC cost Δ, %
1	5	14.48	51.71	1.41	33.96	1.01
2	10	47.46	40.12	0.66	20.63	0.60

3.3.4 Average of top five under distributionally robust exogenous risk

Here we continue with the model above, but, as in Section 3.3.2, we use a multiplicative model for endogenous risk. More precisely, for a tuple $s \in \mathcal{B}^5$ we write

$$\phi_s(x|z) \;=\; \frac{1}{5}\left(\sum_{(k,m)\in s} (1+z_{km})T_{km}\right),$$

where $z_{km} \geq 0$ is the multiplicative exogenous impact on (k,m). Continuing as above, we will now focus on

$$\Phi(x) \;=\; \max_{z\in\mathcal{Z}} \max_{s\in\mathcal{B}^5} \; \phi_s(x|z) \;=\; \max_{z\in\mathcal{Z}} \max_{s\in\mathcal{B}^5} \; \frac{1}{5}\left(\sum_{(k,m)\in s} (1+z_{km})T_{k,m} \right).$$

For these experiments we used

$$\mathcal{Z} \;=\; \{z \in \mathbb{R}^{\mathcal{B}} \,:\, \|z\|_2 \leq 1\},$$

i.e., the Euclidean unit ball. In terms of implementation of Algorithm 1 we used the following methodology:

- The boosting step used the greedy method, that is to say, at iteration t we set

$$z^t \;=\; \operatorname*{argmax}_{\|z\|_2=1} \max_{s\in\mathcal{B}^5} \sum_{(k,m)\in s} z_{km}T_{km} \;=\; \operatorname*{argmax}_{\|z\|_2=1} \sum_{(k,m)\in s^*} z_{km}T_{km},$$

  where $s^* \in \mathcal{B}^5$ is the ordered 5-tuple made up of the top 5 branches under the T_{km} metric. A computation shows that

$$z^t_{k,m} = \frac{T_{km}}{\sqrt{\sum_{(k',m')\in s^*}(T^t_{k',m'})^2}} \text{ for } (k,m) \in s^*, \quad z^t_{k,m} = 0 \text{ otherwise.}$$

- For the cutting step we used the same approach as in the above section, i.e., we select a set of tuples s (including s^*) (a total of ten tuples in the set) and for each tuple we add the cut $\Phi_L \geq \frac{1}{5}\sum_{(k,m)\in s}(1+z^t_{km})T_{km}$.

#	iterations	runtime (s)	risk reduction, %	cost Δ, %	AC risk reduction, %	AC cost Δ, %
1	10	11.92	56.78	1.34	36.26	1.03
2	12	62.66	43.39	0.88	27.20	0.81

3.3.5 Analysis of outcomes

Here we compare the solutions computed using the above models, as specific to case #2 (the 30,000-node system).

In Figure 4 we display the top 30 values T_{km} for the solution computed in the nominal case, the max-robust case in Section 3.3.2, the max-top-10-average in Section 3.3.3 and the average-top-5-robust case considered in Section 3.3.4. It is important to note that the 30 values plotted on each of these four curves refer, in each case, to a set of 30

As expected, all solutions produced by the algorithm substantially reduce the risk present in the nominal solution. The max-robust solution is the least attractive among those but even so, it continues to de-risk the nominal solution even after we move past the maximum.

Another feature of interest is that these three solutions have similar $\max_{km} T_{km}$ metric. However the "max-robust solution" appears inferior to the "average of top 5 (robust)" and the "average of top 10" solutions. The latter, superficially, appears to be the most appealing, but it should be remembered that all three of the "robust" solutions have that additional attribute, i.e., they are all protected against (a model of) exogenous risk.

An additional perspective is gained by comparing how the different solutions arrange power flows. The table below displays the top ten branches in terms of the thermal metric $T_{k,m}$ for the nominal solution:

rank	1	2	3	4	5	6	7	8	9	10
branch	30172	3170	1408	5295	1412	5296	5463	2741	5287	17813
T_{km}	0.62	0.46	0.40	0.37	0.34	0.34	0.31	0.30	0.30	0.27
P_{km}	11.59	38.70	−37.23	−37.89	−32.12	−33.79	−28.44	33.64	28.43	8.63

We can compare these values to the outcome from the "average of top five, robust" case, where we highlight the branches that do not appear in the table above:

rank	1	2	3	4	5	6	7	8	9	10
branch	1408	17813	30172	**28795**	3170	2741	**21388**	**7632**	1412	**17835**
T_{km}	0.33	0.26	0.22	0.22	0.22	0.22	0.22	0.21	0.21	0.21
P_{km}	−33.53	8.51	6.86	−8.61	27.22	28.30	16.83	32.45	−25.28	−21.53

different branches (k,m).

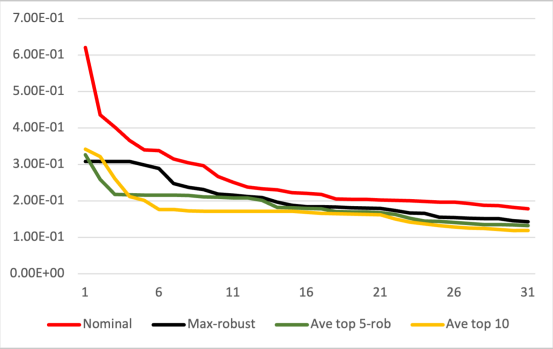

Figure 4: Thermal metrics comparison

Note that the top six T_{km} entries in the table depicting the nominal solution are all higher than the maximum for the robust solution (i.e., 0.33) even though that solution only considers the average among the top five. Moreover, branch 17813 is the tenth-ranked in the nominal solution yet it attains a metric that is higher than all but the highest branch in the robust solution.

References

[1] AMPL Optimization Inc.: AMPL Python API (2026), https://ampl.com, accessed: May 23, 2026

[2] Babaeinejadsarookolaee, S., Birchfield, A., Coffrin, C., et al.: The Power Grid Library for Benchmarking AC Optimal Power Flow Algorithms. arXiv preprint arXiv:1908.02788 (2019)

[3] Beck, Y., Ljubić, I., Schmidt, M.: Linear and Mixed-Integer Bilevel Optimization: Theory and Algorithms. Cambridge University Press (2026), https://yasminebeck.github.io/files/bilevel-optimization-cup.pdf

[4] Benders, J.F.: Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik **4**(1), 238–252 (1962). https://doi.org/10.1007/BF01386316, https://doi.org/10.1007/BF01386316

[5] Bertsimas, D., Brown, D.B., Caramanis, C.: Theory and applications of robust optimization. SIAM Review **53**(3), 464–501 (2011)

[6] Bertsimas, D., Sim, M.: The price of robustness. Operations Research **52**(1), 35–53 (2004)

[7] Bienstock, D.: Electrical transmission system cascades and vulnerability, an Operations Research viewpoint. Society for Industrial and Applied Mathematics (2015)

[8] Bienstock, D., Escobar, M., Gentile, C., Liberti, L.: Mathematical programming formulations for the alternating current optimal power flow problem. 4OR **18**(3), 249–292 (Sep 2020). https://doi.org/10.1007/s10288-020-00455-w, https://link.springer.com/10.1007/s10288-020-00455-w

[9] Bienstock, D.: Potential Function Methods for Approximately Solving Linear Programming Problems: Theory and Practice, vol. 53. Springer Science & Business Media (2006)

[10] Bienstock, D., Iyengar, G.: Solving fractional packing problems in $\mathcal{O}^*(1/\varepsilon)$ iterations. In: Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing. pp. 146–155. STOC '04, Association for Computing Machinery, New York, NY, USA (2004). https://doi.org/10.1145/1007352.1007376

[11] Bienstock, D., Verma, A.: The N-K problem in power grids: New models, formulations, and numerical experiments. SIAM Journal on Optimization **20**(5), 2352–2380 (2010)

[12] Brown, G., Carlyle, M., Salmerón, J., Wood, K.: Defending critical infrastructure. Interfaces **36**(6), 530–544 (2006)

[13] Byrd, R., Nocedal, J., Waltz, R.: Knitro: An Integrated Package for Nonlinear Optimizationİn: Large-Scale Nonlinear Optimization, vol. 83, pp. 35–59. Springer (2006)

[14] Cain, M.B., O'Neill, R.P., Castillo, A.: History of Optimal Power Flow and Formulations. Federal Energy Regulatory Commission (2012)

[15] Carpentier, J.L.: Contribution a l'etude du dispatching economique. Bulletin de la Societe Francoise des Electriciens **8**, 431–447 (1962)

[16] Coffrin, C., Van Hentenryck, P.: A linear-programming approximation of ac power flows. INFORMS Journal on Computing **26**, 718–734 (2014)

[17] Cormican, K.J., Morton, D.P., Wood, R.K.: Stochastic network interdiction. Operations Research **46**(2), 184–197 (1998)

[18] Fratta, L., Gerla, M., Kleinrock, L.: The flow deviation method: An approach to store-and-forward communication network design. Networks **3**(2), 97–133 (1973)

[19] Furini, F., Traversi, E., Belotti, P., Frangioni, A., Gleixner, A., Gould, N., Liberti, L., Lodi, A., Misener, R., Mittelmann, H., Sahinidis, N., Vigerske, S., Wiegele, A.: Qplib: a library of quadratic programming instances. Mathematical Programming Computation **11**(2), 237–265 (Jun 2019). https://doi.org/10.1007/s12532-018-0147-4, publisher Copyright: © 2018,

Springer-Verlag GmbH Germany, part of Springer Nature and The Mathematical Programming Society.

[20] Grigoriadis, M.D., Khachiyan, L.G.: Fast approximation schemes for convex programs with many blocks and coupling constraints. SIAM Journal on Optimization **4**(1), 86–107 (1994)

[21] Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual (2026), https://www.gurobi.com

[22] IEEE Power and Energy Society: IEEE Standard for Calculating the Current-Temperature Relationship of Bare Overhead Conductors. IEEE Standard IEEE Std 738-2012, New York, NY, USA (December 2012). https://doi.org/10.1109/IEEESTD.2012.6401981

[23] Kelley, J.E.J.: The Cutting-Plane Method for Solving Convex Programs. Journal of the Society for Industrial and Applied Mathematics pp. 703–712 (1960)

[24] Kilb, J., Newman, A., Bienstock, D.: Probabilistic Modeling versus Robust Optimization: A tutorial based on a humanitarian logistics use case. arXiv:2604.02493 pp. 1–46 (2026)

[25] Kleinrock, L.: Queueing Systems, Volume 1. John Wiley & Sons, Inc. (1975)

[26] Martin, B. and Willis, H. and Strong, A.: Managing Systemic Supply Chain Risk to the U.S. Economy from Trade Concentration and Geopolitical Conflict: The Roles of Insurance and Other Hedging Strategies (2026), https://www.rand.org/pubs/research_reports/RRA4291-1.html

[27] Misener, R., Floudas, C.: Advances for the pooling problem: Modeling, global optimization, and computational studies survey. Applied and Computational Mathematics **8** (01 2009)

[28] Molzahn, D.K., Hiskens, I.A.: A Survey of Relaxations and Approximations of the Power Flow Equations. Now, Foundations and Trends (2019)

[29] Nocedal, J., Wright, S.: Numerical Optimization. Springer Series in Operations Research and Financial Engineering, Springer, 2 edn. (2006)

[30] Plotkin, S.A., Shmoys, D.B., Tardos, É.: Fast approximation algorithms for fractional packing and covering problems. In: 32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991. pp. 495–504. IEEE Computer Society (1991). https://doi.org/10.1109/SFCS.1991.185411

[31] Salmerón, J., Wood, K., Baldick, R.: Worst-Case Interdiction Analysis of Large-Scale Electric Power Grids. IEEE Trans. Power Systems **24**, 96–104 (2009)

[32] Shahrokhi, F., Matula, D.W.: The maximum concurrent flow problem. J. ACM **37**, 318–334 (1990), https://api.semanticscholar.org/CorpusID:4469579

[33] Sisson, B., Bienstock, D.: Polishing solutions to nonlinear optimization problems. forthcoming (2026)

[34] Vorobyov, S., Gershman, A., Luo, Z.Q.: Robust adaptive beamforming using worst-case performance optimization: a solution to the signal mismatch problem. IEEE Transactions on Signal Processing **51**(2), 313–324 (2003). https://doi.org/10.1109/TSP.2002.806865

[35] Zeiler, M.D.: ADADELTA: An adaptive learning rate method. arXiv preprint arXiv:1212.5701 (2012)

[36] Zimmerman, R., Murillo-Sanchez, C., Gan, D.: MATPOWER, A MATLAB Power System Simulation Package. IEEE Trans. Power Sys. **26**, 12–19 (2011)