SIAM J. OPTIM.
Vol. 28, No. 2, pp. 1726–1750

CARDINALITY OF UPPER AVERAGE AND ITS APPLICATION TO NETWORK OPTIMIZATION*

MATTHEW NORTON†, ALEXANDER MAFUSALOV‡, AND STAN URYASEV§

Abstract. We propose a new characteristic for counting the number of large outcomes in a data set that are considered to be large with respect to some fixed threshold x. A popular characteristic used for this purpose is the Cardinality of Upper Tail (CUT), which counts the number of outcomes with magnitude larger than the threshold. We propose a similar characteristic called the Cardinality of Upper Average (CUA), defined as the number of largest data points which have average value equal to the threshold. CUA not only assesses the number of outcomes that are large, but also their overall magnitude. CUA also has superior mathematical properties: it is a continuous function of the threshold, its reciprocal is piecewise linear with respect to threshold, and it is directly optimizable via convex and linear programming. This is in contrast to CUT, which does not asses the severity of large outcomes, is discontinuous as a function of threshold, and is such that direct optimization yields numerically difficult nonconvex problems. We show that CUA can be used to formulate meaningful optimization problems containing counters of the largest components of a vector without introduction of binary variables, leading to large improvement in computation speeds. In particular, we apply the CUA concept to create new formulations of network optimization problems involving overloaded nodes or edges, where we aim to minimize the number of most burdened nodes or edges.

Key words. cardinality of upper average, buffered probability of exceedance, network optimization, conditional value-at-risk, mixed integer programming, linear programming

AMS subject classifications. 90C05, 90C35, 90C25, 90C11, 90C90

DOI. 10.1137/16M1095913

1. Introduction.

When analyzing a set of n data points, one often needs to count the number of outcomes that are large compared to some threshold $x \in \mathbb{R}$. For example, if the data points represent monetary losses, it is only natural that one would want to count the number of outcomes that are large compared to a threshold representing an "acceptable" level of loss. A popular characteristic which acts as a counter of such large outcomes is the x-Cardinality of Upper Tail (CUT_x), which counts the number of data points with magnitude exceeding the threshold x. We introduce a similar characteristic called the x-Cardinality of Upper Average (CUA_x), defined as the number of largest data points which have average value equal to the threshold x. Thus, CUA_x not only counts the number of data points with magnitude larger than the threshold, but also the largest outcomes with magnitude less than the threshold such that the average of these outcomes is equal to x.

Consider the following example to illustrate the conceptual difference between CUA_x and CUT_x. Suppose that we have 10,000 pieces of gold. We want to make

*Received by the editors September 29, 2016; accepted for publication (in revised form) February 12, 2018; published electronically June 5, 2018.
http://www.siam.org/journals/siopt/28-2/M109591.html
Funding: This work was partially supported by the USA AFOSR grants "Design and Redesign of Engineering Systems," FA9550-12-1-0427, and "New Developments in Uncertainty: Linking Risk Management, Reliability, Statistics and Stochastic Optimization," FA9550-11-1-0258.
†Operations Research Department, Naval Postgraduate School, Monterey, CA 93943 (mnorton@nps.edu).
‡Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL 32611 (mafusalov@ufl.edu).
§Director of Risk Management and Financial Engineering Lab, Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL 32611 (uryasev@ufl.edu).

some statement about the number of heavy pieces of gold in this set because we know, for instance, that it takes 10 grams of gold to make one gold coin. So, 10 grams can be considered a natural reference point (or threshold) for evaluating the heaviness of gold pieces. After analyzing the dataset, we learn that there are only 3 pieces of gold with weight exceeding 10 grams. This means that $\mathrm{CUT}_{10} = 3$ and, thus, that at least 3 gold coins can be made from the 3 heaviest pieces. Suppose, though, that the weight of the largest piece of gold equals 1,000 grams. Then, the lower bound, 3 coins, is quite far away from the number of coins which can be made from the heaviest 3 pieces of gold. We can make at least 102 coins (i.e., 100 coins from the largest piece plus two more coins from the other two heaviest pieces). CUT_x does not provide information about the heaviness of the largest pieces, only the lower bound for the individual pieces. Now, suppose we have calculated that $\mathrm{CUA}_{10} = 207.3$. This actually means that you can make 207.3 coins from the heaviest 207.3 pieces. This is the case because the average weight of a gold piece in the selected subset equals 10 grams. Also, notice that CUA_{10} tells us that the 208th piece needs to be cut! CUA_x is a continuous (nice mathematical) characteristic as a function of the threshold (10 grams), while CUT_x is a discontinuous characteristic. CUA_x provides information about the average heaviness of largest pieces.

This paper is focused on logistics applications, and therefore let us discuss again the same example in a logistics context. Suppose that we manage a network with 10,000 servers and assume that a server is "overloaded" if the length of the queue is greater than or equal to 10 jobs. We can ask the question, How many servers are overloaded? The answer is $\mathrm{CUT}_{10} = 3$. However, it is unclear how significantly these servers are overloaded and what additional resources would be required to serve the customers in these queues. An alternative characteristic is $\mathrm{CUA}_{10} = 207.3$. This means that the average length of a queue over the 207.3 busiest servers is 10 jobs and the total number of unserved jobs in these servers is $207.3 \times 10 = 2{,}073$. Therefore, we understand immediately the total number of unserved clients in the 203.7 longest queues. This important information is present in CUA_x and is not present in CUT_x.

You may ask, What is a better characteristic, CUA_x or CUT_x? The simple answer is that these are different, complimentary characteristics. Therefore, while in some cases you may be interested in CUT_x and in other cases CUA_x, it is probably a good idea to calculate both characteristics.

In the most basic sense, CUA_x is similar to CUT_x, with both acting as counters of large outcomes relative to a threshold. Beyond this, they have important differences. First, as illustrated in the example, CUA_x considers how far the outcome magnitudes are from the threshold. The CUT_x characteristic only considers if an outcome is larger than x, not how far beyond the threshold these magnitudes lie. Second, CUA_x has superior mathematical properties. It is continuous w.r.t. the threshold parameter and a partial derivative can be taken w.r.t. threshold yielding information regarding the sensitivity of CUA_x to threshold changes. It is also piecewise linear in its reciprocal. As we will show, this allows for efficient calculation of CUA_x for all thresholds $x \in \mathbb{R}$.

In the context of optimization, the properties of CUA_x yield substantial benefits, with direct optimization of CUA_x reducing to convex, sometimes linear, programming. This means that we are able to efficiently minimize the number of tail outcomes, while taking into account the severity of these outcomes. Furthermore, the tail that is minimized is determined by the threshold, with the tail including the largest outcomes that average to the specified threshold. This is all in contrast to CUT_x, which does not account for the severity of the tail outcomes, is discontinuous w.r.t. the threshold parameter, and is such that direct optimization yields numerically challenging nonconvex problems.

While being different from CUT_x in many ways, CUA_x is uniquely related. We show that CUA_x is, in a certain sense, the minimal quasiconvex upper bound of CUT_x. In an optimization context, this could be useful, suggesting that CUA_x minimization may yield efficient heuristic approaches to reducing CUT_x. This, though, is not our focus and we emphasize that CUA_x and its optimization are meaningful in their own right.

We apply CUA_x to a specific optimization setting, providing convex and Linear Programming (LP) CUA_x minimization formulations for solving network optimization problems, specifically addressing variants of the generalized assignment, capacity planning, and min cost network flow problems. In general, we consider network flow problems where product must flow through a network and we assume that intermediate nodes or arcs can become "overloaded" if too much flow is pushed through them. Therefore, it is desirable to minimize the number of overloaded nodes or arcs. We formulate this as a CUA_x minimization problem, which accounts for the magnitude of loads put upon nodes/arcs in the overloaded state, as well as the nodes/arcs with loads less than, but most near to, the overloaded state. This information may be meaningful, as it may be undesirable to have nodes/arcs that, while not yet overloaded, are close to the overloaded state. Furthermore, it may be undesirable to have nodes/arcs that are dramatically overloaded. Thus, we minimize the number of most overloaded nodes/arcs that have average load equal to x, where nodes/arcs with loads larger than x are considered to be overloaded.

We compare these CUA_x formulations with similar formulations that minimize CUT_x, directly minimizing the number of nodes/arcs in the overloaded state. With this problem involving binary variables to indicate the state of the node/arc, Mixed Integer Programming (MIP) must be used. Not only is this much more difficult to solve than the LP CUA_x minimization, but we show that it may provide less appealing policies. By minimizing CUT_x, information about the magnitude of the loads is ignored and it only considers whether a node/arc is overloaded or not. This information, as already mentioned, can be meaningful. Thus, the CUA_x formulation may be more appropriate and suggest more appealing policies.

The CUA_x concept is actually a deterministic variant of so called Buffered Probability of Exceedance (bPOE). A detailed discussion of this concept, studied in [9, 12, 16, 17, 18], is beyond the scope of this paper. We maintain a deterministic setting, while bPOE is studied in a probabilistic, stochastic optimization setting. We do, though, include some background connecting CUA_x and bPOE in Appendix A for the interested reader.

This paper is organized as follows. Section 2 discusses the task of analyzing the tail of a data distribution in a deterministic setting, which serves to set the stage for defining CUA_x. Section 3.1 defines CUA_x. We show that CUA_x is efficient to calculate and give an example illustrating CUA_x, particularly as an upper bound of CUT_x. Additionally, we provide an efficient method of calculating CUA_x which utilizes the piecewise linearity of its reciprocal. Section 3.2 provides relations between CUA_x and CUT_x, showing that it is possible to simultaneously calculate CUA_x and CUT_x. Section 4 shows the power of the CUA_x concept when applied to an optimization setting. Specifically, we discuss classes of network optimization problems that are traditionally formulated as MIP problems and show how an application of CUA_x leads to convex or linear programming reformulations of analogous problems.

2. Cardinality of upper tail and related quantities.

2.1. Cardinality of upper tail and upper average. Consider a Euclidean vector $\mathbf{y} = (y_1, \dots, y_n)$ containing n data points. It is often important in applications to know the number of components of $\mathbf{y}$ that exceed a particular threshold $x \in \mathbb{R}$. We call this the Cardinality of the Upper Tail (CUT_x), denoted by

$$\mathrm{CUT}_x(\mathbf{y}) = \eta_x(\mathbf{y}) = |\{y_i | y_i \geq x, i = 1, \dots, n\}| \ .$$

This quantity, though, can be difficult to work with. For example, it is discontinuous w.r.t. the x parameter. Additionally, this quantity does not provide information about the magnitude of the components above the threshold x. As we will show in later sections, these are undesirable properties in an optimization setting. We can also consider other quantities that provide useful information about the tail of the data distribution that are easy to work with. For example, one could consider the ordered weighted averaging aggregation operators of [19]. We can also consider a type of *tail average* called the k-Upper Average (UA_k), which provides information about the magnitude of data points in the tail and upper bounds on CUT_x. If we let $(y^{(1)}, y^{(2)}, \dots, y^{(n)})$ represent a permutation of $(y_1, \dots, y_n)$ with components listed in nondecreasing order, $y^{(1)} \leq y^{(2)} \leq \cdots \leq y^{(n)}$, we can denote this quantity by

$$\mathrm{UA}_k(\mathbf{y}) = \frac{1}{k} \sum_{i=n-k+1}^{n} y^{(i)} \ . \tag{1}$$

2.2. Generalized $\mathbf{UA}_k$. Notice that (1) only applies to integer values of the k parameter. This function, $\mathrm{UA}_k(\cdot)$, can also be defined in a more general manner for noninteger values of k. Popularized in the financial engineering literature under the name CVaR, the paper [15] defines this quantity in a broad, probabilistic setting. Here, we maintain a deterministic setting, but emphasize that this is simply a special case of the general CVaR definition and formula. Thus, following directly from [15], we have that for any $k \in [1, n]$, letting $[\cdot]^+ = \max\{0, \cdot\}$,

$$\mathrm{UA}_k(\mathbf{y}) = \min_{\gamma} \left\{ \gamma + \frac{1}{k} \sum_{i=1}^{n} [y_i - \gamma]^+ \right\} \ . \tag{2}$$

Though this formula may not seem intuitive, it can be clarified by noticing two facts. First, the optimal objective value of (2) is equal to (1) as long as $k \in \{1, \dots, n\}$ is an integer, meaning that for the integer case, the seemingly complex formula of (2) simply yields the average of the k largest components. Second, for noninteger values of $k \in [1, n]$, this formula simply gives a weighted average of $\mathrm{UA}_{\lfloor k \rfloor}(\mathbf{y})$ and $\mathrm{UA}_{\lceil k \rceil}(\mathbf{y})$, where $\lfloor k \rfloor$ denotes the largest integer less than or equal to k and $\lceil k \rceil$ denotes the smallest integer greater than or equal to k. Thus, intuitively, formulation (2) is still averaging the largest k data points, but it is now a continuous function w.r.t. the k parameter.

In addition to providing useful information about the magnitude of data points in the tail, $\mathrm{UA}_k(\cdot)$ provides an upper bound for CUT_x. Specifically, we have the following relation, which follows intuitively from (1) and the definition of CUT_x:

$$\mathrm{UA}_k(\mathbf{y}) = x \implies \eta_x(\mathbf{y}) \leq k \ . \tag{3}$$

In this paper, we define CUA_x, the inverse of (2). As we will show, CUA_x can be efficiently calculated and provides valuable information about the largest components of our data vector $\mathbf{y}$. Additionally, CUA_x can be efficiently optimized with convex and linear programming. We also show that CUA_x can be used to formulate optimization problems that are similar to, yet fundamentally different to CUT_x optimization problems. We show that CUA_x optimization can sometimes be more appropriate and may suggest more appealing optimal policies than the similar CUT_x minimization problems.

3. Cardinality of the upper average.

3.1. Definition of $\mathbf{CUA}_{x}$.

For a specified threshold x, CUA_x calculates the value of $k \in [1, n]$ such that $\mathrm{UA}_k(\mathbf{y}) = x$. In words, CUA_x is equal to the number of largest components of the vector $\mathbf{y}$ such that the average of those components is equal to x. We now present Theorem 1, which provides two equivalent ways of defining CUA_x. While (5) could be viewed as the more intuitive of the two, we focus on the more tractable representation (4) throughout this paper. We then provide examples illustrating key differences between CUT_x and CUA_x.

THEOREM 1. *Consider a Euclidean vector* $\mathbf{y} = (y_1, \dots, y_n)$. CUA_x *of* $\mathbf{y}$ *at threshold* $x \in \mathbb{R}$ *is defined as*

$$\bar{\eta}_x(\mathbf{y}) = \min_{a \geq 0} \sum_{i=1}^{n} [a(y_i - x) + 1]^+ \tag{4}$$

and can also be represented as

$$\bar{\eta}_x(\mathbf{y}) = \begin{cases} \max\{k | \frac{1}{k}(\sum_{i=1}^{\lfloor k \rfloor} y^{(n-i+1)} + (k - \lfloor k \rfloor) y^{(n-\lceil k \rceil+1)}) \geq x\} & \text{if } x \leq \max_i y_i \,, \\ 0 & \text{otherwise} \,. \end{cases} \tag{5}$$

Proof. Here we show that for a Euclidean vector $\mathbf{y} \in \mathbb{R}^n$, equation (4) given for CUA_x at threshold $x \in \mathbb{R}$ equals the value of k such that $\mathrm{UA}_k(\mathbf{y}) = x$. We must address four cases. The equivalence of (4) and (5) follows from these cases.

Case 1. Assume that $x \in (\mathrm{UA}_n(\mathbf{y}), \mathrm{UA}_1(\mathbf{y}))$. This assumption ensures that there exists a value of $k \in [1, n]$ such that $\mathrm{UA}_k(\mathbf{y}) = x$. We want to find the value of k such that $\mathrm{UA}_k(\mathbf{y}) = x$, so we write CUA_x as

$$\bar{\eta}_x(\mathbf{y}) = \{k | \mathrm{UA}_k(\mathbf{y}) = x\} \,. \tag{6}$$

Notice now that $\mathrm{UA}_k(\mathbf{y})$ is a strictly increasing function of k on $k \in [\frac{n-m}{n}, n]$, where $m = |\{y_i | y_i = \max_i y_i\}|$ (i.e., m equals the number of components of $\mathbf{y}$ that are equal to the largest component of $\mathbf{y}$). This follows from the known result (see [15]) that CVaR is strictly increasing on an equivalent interval. Because of this, we see that (6) can be rewritten as the unique solution to

$$\min\{k | \mathrm{UA}_k(\mathbf{y}) \leq x\} \,. \tag{7}$$

Substituting (2) for $\mathrm{UA}_k(\mathbf{y})$, equation (7) becomes

$$\bar{\eta}_x(\mathbf{y}) = \min \left\{ k \,\middle|\, \min_{\gamma} \gamma + \frac{1}{k} \sum_{i=1}^{n} [y_i - \gamma]^+ \leq x \right\} \,. \tag{8}$$

This can then be simplified as

$$\begin{aligned} \bar{\eta}_x(\mathbf{y}) = \min_{k,\gamma} \quad & k \\ \text{s.t.} \quad & \gamma + \frac{1}{k} \sum_{i=1}^{n} [y_i - \gamma]^+ \leq x \,, \end{aligned} \tag{9}$$

where "s.t." indicates "subject to." Explicitly enforcing the constraint $x - \gamma > 0$, we can further simplify (9) without changing the optimal solution to become

$$\bar{\eta}_x(\mathbf{y}) = \min_{k, x-\gamma>0} \quad k \qquad \text{s.t.} \quad \frac{\sum_{i=1}^n [y_i - \gamma]^+}{x-\gamma} \leq k \;. \tag{10}$$

This, though, can be further simplified to become

$$\bar{\eta}_x(\mathbf{y}) = \min_{x-\gamma>0} \quad \frac{\sum_{i=1}^n [y_i - \gamma]^+}{x-\gamma} \;. \tag{11}$$

Finally, with the change of variable $a = \frac{1}{x-\gamma}$, (11) can be transformed into

$$\bar{\eta}_x(\mathbf{y}) = \min_{a\geq 0} \quad \sum_{i=1}^n [a(y_i - x) + 1]^+ \;. \tag{12}$$

Case 2. Assume that $x = \max_j y_j$. With $x = \max_j y_j$, we show that CUA_x equals the number of components equal to $\max_j y_j$. For notational convenience let $\max_j y_j = y_{max}$. Also, assume there are m components of $\mathbf{y}$ that are equal to y_{max}, i.e., $|\{y_i | y_i = y_{max}\}| = m$. Also, let $\hat{y} = \max\{y_i | y_i < y_{max}\}$, i.e., $\hat{y}$ equals the largest component of $\mathbf{y}$ that is less than y_{max}.[1]

Since $y_j - y_{max} \leq 0$ for any $j \in \{1, \ldots, n\}$, we have that, for any $a^* \geq \frac{-1}{\hat{y} - y_{max}}$,

$$\begin{aligned} \min_{a\geq 0} \sum_{i=1}^n [a(y_i - y_{max}) + 1]^+ &\geq \min_{a\geq 0} \sum_{y_i = y_{max}} [a(y_i - y_{max}) + 1]^+ \\ &= \sum_{i=1}^n [a^*(y_i - y_{max}) + 1]^+ = |\{y_i | y_i = y_{max}\}| = m \;. \end{aligned}$$

To see this, notice that for any $y_i \leq \hat{y}$ and any $a^* \geq \frac{-1}{\hat{y} - y_{max}}$ we get

$$[a^*(y_i - y_{max}) + 1]^+ \leq [a^*(\hat{y} - y_{max}) + 1]^+ = [-1 + 1]^+ = 0 \;.$$

Furthermore, for any $y_j = y_{max} = x$, we have that

$$[a(y_i - y_{max}) + 1]^+ = [a(0) + 1]^+ = 1 \;.$$

Case 3. Assume that $x > \max_i y_i$. We need to show that (4) equals the value of k such that $\text{UA}_k(\mathbf{y}) = x$. Since $x > \max_i y_i$ we show that CUA_x equals 0.

Let us write $\max_j y_j = y_{max}$. Since $x > y_{max}$, for any $i \in \{1, \ldots, n\}$ we have $y_i - x < 0$. This implies that for any $a^* \geq \frac{-1}{y_{max} - x}$, $a^*(y_i - x) \leq -1$ for all i, and therefore

$$\sum_{i=1}^n [a^*(y_i - x) + 1]^+ = 0 = \min_{a\geq 0} \sum_{i=1}^n [a(y_i - x) + 1]^+ \;.$$

Case 4. Assume that $x \leq \text{UA}_n(\mathbf{y})$. We need to show that (4) equals the value of k such that $\text{UA}_k(\mathbf{y}) = x$. Since $x \leq \text{UA}_n(\mathbf{y})$, we show that CUA_x equals n.

[1] If all components of $\mathbf{y}$ are equal, then apply Case 4. In this case, although $\hat{y}$ does not exist, $\text{UA}_n(\mathbf{y}) = \max_j y_j$ and Case 4 can be applied.

Since $x \le \mathrm{UA}_n(\mathbf{y})$, we have $0 \le \mathrm{UA}_n(\mathbf{y}) - x$. This implies that, for any $a \ge 0$,

$$\sum_{i=1}^n [a(y_i - x) + 1]^+ \ge \sum_{i=1}^n [a(y_i - x) + 1] \ge n[a(\mathrm{UA}_n(\mathbf{y}) - x) + 1] \ge n\ .$$

This result implies that $\min_{a \ge 0} \sum_{i=1}^n [a(y_i - x) + 1]^+ = n$ (attained at $a = 0$). ◻

Initially, it may be surprising that this particular formula calculates the number of biggest components of the vector $\mathbf{y}$ such that the average of those components is equal to x. In short, this formula is derived from (2), partially explaining its form as the unique minimal value to the minimization problem. Note that the proof of Theorem 1 also shows how CUA_x is defined for thresholds $x \notin (\mathrm{UA}_n(\mathbf{y}), \mathrm{UA}_1(\mathbf{y}))$, where $\mathrm{UA}_n(\mathbf{y}) = \frac{1}{n}\sum_{i=1}^n y_i$ and $\mathrm{UA}_1(\mathbf{y}) = \max_i y_i$. Furthermore, the definition of CUA_x for these extreme cases is motivated by its connection to bPOE, a new concept recently studied in [9, 12, 16, 17, 18]. Appendix A includes a brief discussion of this connection showing that CUA_x can be viewed as a deterministic variant of upper bPOE [12, 9]. A detailed discussion of this, though, is beyond the scope of this paper.

To begin discussing CUA_x, first note that CUA_x is continuous w.r.t. the parameter x on the interval $x \in (-\infty, \mathrm{UA}_1(\mathbf{y}))$; this will be shown later on in Corollary 1. As already mentioned in section 2, CUT_x is discontinuous w.r.t. this threshold parameter. Second, notice that by calculating the $k \in [1, n]$ such that $\mathrm{UA}_k(\mathbf{y}) = x$, CUA_x is counting all components with magnitude greater than x and some components with magnitude less than x. These magnitudes can contain important and meaningful information that is ignored by CUT_x.

For example, assume you are deciding whether or not to place a service facility in a particular location (e.g., a cell phone tower), with the service facility being able to serve only houses within a geographic radius of R miles. Let $\mathbf{y}$ then represent the entire set of customers you wish to serve from this location and their distances from the facility. To assess the quality of this location, it is intuitive to look at $\mathrm{CUT}_R(\mathbf{y})$, which gives you the number of customers within this set that will go unserved by this facility. However, the use of such a *hard* threshold is quite nonintuitive under closer inspection. Decision makers would certainly like to know if a large number of customers are located $R + \epsilon$ miles away from the facility, where ϵ is a very small number. Similarly, they would also like to know if a large number of customers are $R - \epsilon$ miles away. These characteristics are critically important to assessing the quality of this facility and the service it can provide to a set of desired customers. In this example, CUA_x can be an important counterpart to CUT_x, acting as a *soft* threshold, counting the number of customers *around* R miles away from the facility.

Consider also a disaster relief agency that is analyzing historical hurricane damage data, where damages are in dollars [7]. Assume that the agency is attempting to determine if allocating \$$B$ dollars to the relief fund for the next hurricane is sufficient. First, of the historical damage amounts that exceed \$$B$, it is important to know by how much these damages exceeded \$$B$, especially if the exceedance is large. Secondly, it is important to know the magnitude of the largest damage amounts that are less than \$$B$. Are these damages very close to \$$B$? Or are they much smaller than \$$B$? Answers to these questions are clearly important for budgetary considerations.

Next, consider the simple, illustrative example in Figure 1 with data vector $\mathbf{y} = (1, 2, 5, 7)$. For this vector, $\mathrm{CUA}_x = 2$ for $x = 6$, because the average of the two largest components of the vector $\mathbf{y}$ equals $(7 + 5)/2 = 6$, and $\mathrm{CUA}_x = 3$ for $x = 4\frac{2}{3} = (2 + 5 + 7)/3$. We can observe that CUA_x is an upper bound for CUT_x. Additionally, for this example, Figure 1 shows that CUA_x is continuous w.r.t. x except at the

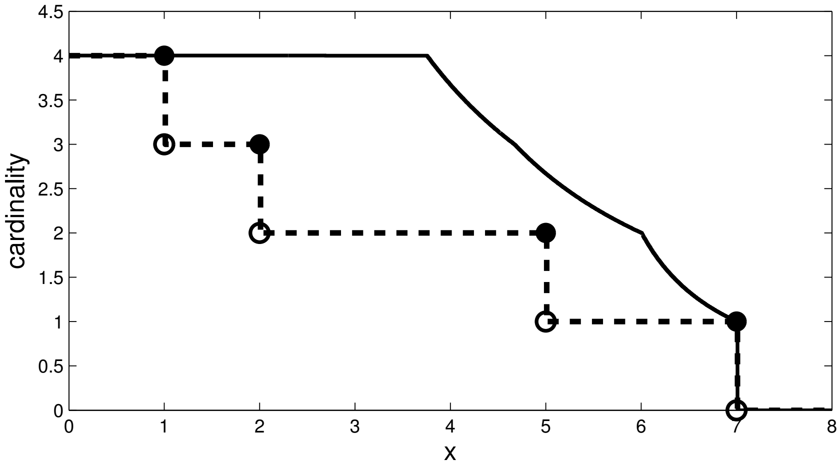

FIG. 1. *Cardinality of upper tail* (CUT_x)*,* $\eta_x(\mathbf{y})$ *(dashed line) and cardinality of upper average* (CUA_x)*,* $\bar{\eta}_x(\mathbf{y})$ *(solid line) as functions of threshold* x*, where* $\mathbf{y} = (1, 2, 5, 7)$*.*

maximum point where threshold $x = 7$, while CUT_x is discontinuous at $x = 1, 2, 5$, and 7.

With CUA_x being the inverse of UA_k, both quantities relay similar information. We mention, though, that CUA_x may be more intuitive to use when the data have meaningful units, particularly because the threshold parameter x is posed in the associated units. For example, if the data represent monetary losses (in the unit of dollars) from hurricane damage, it can be more intuitive to work with a threshold x, which will be some dollar amount, rather than with the k-parameter. Consider, as before, the disaster relief agency that is attempting to determine if allocating $\$B$ dollars to the relief fund for the next hurricane is sufficient. It would be much more intuitive to look at CUA_B than to look at UA_k for many different values of k. Additionally, this also applies when considering some optimization tasks. Assume an investor is trying to form a portfolio by analyzing historical stock behavior (where each possible portfolio generates some data set of historical losses it would have incurred in the market). If an investor does not want to incur losses larger than $\$B$ dollars, it is more intuitive to try and devise a portfolio that minimizes CUA_B rather than to find a portfolio that minimizes UA_k for some appropriate value of k which would need to be determined.

We also highlight the simple but useful fact that CUA_x tells you immediately the total weight that is in the tail. Consider again the example from the introduction where we have a network of 10,000 servers and we find that $\mathrm{CUA}_{10} = 207.3$. CUA_x immediately tells us that 10×207.3 additional units of resources are needed to process jobs in the 207.3 longest queues. On the other hand, $\mathrm{CUT}_{10} = 3$ is much less informative, telling us only that the three longest queues need *at least* 30 additional units of resources, even though this number could be much larger. We can put this more precisely, though. Assume that a^* is an optimal point for CUA_x calculation (4) at threshold x. We prove in section 3.2.2 that

$$x\bar{\eta}_x(\mathbf{y}) \leq \sum_i \left\{ y_i \middle| y_i \geq x - \frac{1}{a^*} \right\} \leq x(\lfloor \bar{\eta}_x(\mathbf{y}) \rfloor + 1) \,,$$

where $\lfloor k \rfloor$ denotes the largest integer less than or equal to k and $\lceil k \rceil$ denotes the smallest integer greater than or equal to k. Therefore, we know that the total amount of resources needed to process all unserved jobs waiting in queues with length longer than or equal to $x - \frac{1}{a^*}$ is somewhere between $x\bar{\eta}_x(\mathbf{y})$ and $x(\lfloor \bar{\eta}_x(\mathbf{y}) \rfloor + 1)$. Additionally, note that the continuity of CUA_x can help determine where these resources are needed. For example, consider the case in which all queues have load less than 10 except for three servers, which are extremely overloaded. In this case, one can take a partial derivative of CUA_x w.r.t. the threshold to analyze how dramatically CUA_x is changing as the threshold changes continuously.

3.1.1. $\mathbf{CUA}_x$ calculation via linear interpolation. Suppose that one would like to calculate CUA_x for multiple threshold levels $x \in \mathbb{R}$. One could utilize (4) to achieve this task, but this proves quite inefficient if one would like to know CUA_x for all thresholds $x \in \mathbb{R}$. When one is simply interested in calculating CUA_x for all thresholds $x \in \mathbb{R}$ for a fixed vector $\mathbf{y}$, it is possible to utilize linear interpolation to calculate CUA_x at all thresholds $x \in \mathbb{R}$. To show this, we first introduce the following alternative calculation formula for CUA_x, which shows that CUA_x can be calculated via minimization over a finite set of points. It also provides additional insight into formula (4).

PROPOSITION 1. *Consider a Euclidean vector* $\mathbf{y} = (y_1, \ldots, y_n)$ *and let* $y_0 = -\infty$. CUA_x *of* $\mathbf{y}$ *at threshold* $x \in \mathbb{R}$ *equals*

$$\bar{\eta}_x(\mathbf{y}) = \min_{j \in \{0,1,\ldots,n\}} \frac{\sum_{i=1}^{n} [y_i - y_j]^+}{[x - y_j]^+} \,. \tag{13}$$

Proof. This result follows from a similar result in [9] regarding bPOE in a probabilistic setting. For the interested reader, a simplified deterministic proof can be found in the extended version of this paper.[2] ∎

Utilizing this proposition, Corollary 1 shows that CUA_x can be calculated via simple linear interpolation. Proposition 1, by itself, though, provides interesting insights. First, we see that calculation can be performed by only considering the finite set of points $(y_0, y_1, \ldots, y_n)$. Second, it follows from Proposition 1 that if y_j is the argmin of (13), then $a^* = 1/(x - y_j)$ is an argmin of (4). This fact can be seen more clearly in the proof of Proposition 1 (i.e., see the change of variable that takes place).

Moving to the discussion of linear interpolation, suppose we have the vector $\mathbf{y} \in \mathbb{R}^n$. Instead of using (4) to calculate CUA_x, one only needs to calculate $\mathrm{UA}_k(\mathbf{y})$ for $k \in \{1, \ldots, n-1\}$, which effectively calculates CUA_x for thresholds $x \in \{\mathrm{UA}_{n-1}(\mathbf{y}), \ldots, \mathrm{UA}_1(\mathbf{y})\}$, then utilize linear interpolation to calculate CUA_x for the intermediate threshold values.

COROLLARY 1. *The function* $\frac{1}{\bar{\eta}_x(\mathbf{y})}$ *is a piecewise-linear convex function of* x *with knots at* $x \in \{\mathrm{UA}_n(\mathbf{y}), \mathrm{UA}_{n-1}(\mathbf{y}), \ldots, \mathrm{UA}_1(\mathbf{y})\}$. *Specifically, for any threshold* $x = \lambda \mathrm{UA}_{i+1}(\mathbf{y}) + (1-\lambda)\mathrm{UA}_i(\mathbf{y})$, $i \in \{1, \ldots, n-1\}$, $\lambda \in (0,1)$, *we have that*

$$\frac{1}{\bar{\eta}_x(\mathbf{y})} = \frac{\lambda}{\bar{\eta}_{\mathrm{UA}_{i+1}(\mathbf{y})}(\mathbf{y})} + \frac{1-\lambda}{\bar{\eta}_{\mathrm{UA}_i(\mathbf{y})}(\mathbf{y})} \,. \tag{14}$$

Proof. This result follows from a similar result in [9] regarding bPOE in a probabilistic setting. For the interested reader, a simplified deterministic proof can be found in the extended version of this paper.[3] ∎

[2] See www.ise.ufl.edu/uryasev/publications/.
[3] See www.ise.ufl.edu/uryasev/publications/.

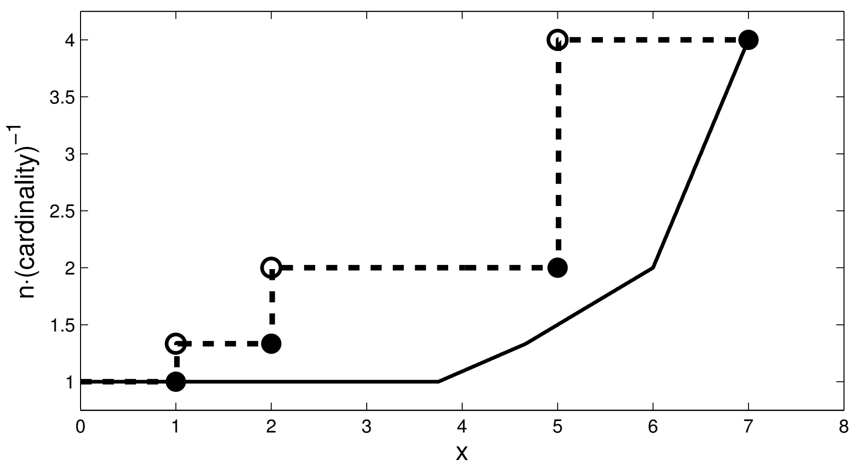

FIG. 2. $\frac{1}{\eta_x(\mathbf{y})}$ *(dashed line) and* $\frac{1}{\bar{\eta}_x(\mathbf{y})}$ *(solid line) as functions of threshold* x*, where* $\mathbf{y} = (1, 2, 5, 7)$.

Therefore, we only need to calculate CUA_x for $n - 1$ thresholds, namely $x \in \{\text{UA}_{n-1}(\mathbf{y}), \ldots, \text{UA}_1(\mathbf{y})\}$, and we can "fill in" the missing thresholds via linear interpolation. Using the example $\mathbf{y} = (1, 2, 5, 7)$ from section 3.1, we illustrate the piecewise linearity in Figure 2, which plots $\frac{1}{\bar{\eta}_x(\mathbf{y})}$ on the vertical axis and x on the horizontal axis.

3.2. Connecting $\mathbf{CUA}_x$ and $\mathbf{CUT}_x$. In this section, we discuss the connection between CUA_x and CUT_x. We emphasize, though, that the core value of CUA_x is not rooted in its relationship with CUT_x. As already mentioned, CUA_x is a complimentary characteristic to CUT_x that is useful in its own right, as it considers information about the magnitude of data points around the threshold, which CUT_x does not, and is a continuous function w.r.t. threshold. In section 4, we discuss this further, showing that CUA_x optimization problems are interesting in their own right, and that their usefulness is not confined by their relationship with CUT_x minimization problems.

3.2.1. $\mathbf{CUA}_x$ as upper bound on $\mathbf{CUT}_x$. Figure 1 shows for a specific example that CUA_x acts as an upper bound for CUT_x. Specifically, this can be posed as the following relation, which follows intuitively from the definitions of CUA_x, UA_k, and CUT_x:

$$\bar{\eta}_x(\mathbf{y}) = k \iff \text{UA}_k(\mathbf{y}) = x \implies \eta_x(\mathbf{y}) \leq k \,. \tag{15}$$

We can improve upon this notion of an upper bound, though, showing that it can be viewed as the minimal quasiconvex upper bound. This is shown in the following proposition, which is a Euclidean space alternative of a statement proved in [9]. An analogous statement for value-at-risk and conditional-value-at-risk can be found in [8]. We call a function on $\mathbb{R}^n$ symmetric if permutation of components of a vector does not change the value of the function. A function g is called quasiconvex if its level-sets $\{\mathbf{y}|g(\mathbf{y}) \leq c\}$ are convex for all c or, equivalently, $g(\lambda\mathbf{y} + (1-\lambda)\mathbf{z}) \leq \max\{g(\mathbf{y}), g(\mathbf{z})\}$ for all $\lambda \in (0, 1)$ and $\mathbf{y}, \mathbf{z} \in \mathbb{R}^n$.

Proposition 2. *Consider a function $g : \mathbb{R}^n \to \mathbb{R}$ which is symmetric, quasiconvex, and greater than η_x everywhere on $\mathbb{R}^n$. Then $g(\mathbf{y}) \geq \lfloor \bar{\eta}_x(\mathbf{y}) \rfloor$ for all $\mathbf{y} \in \mathbb{R}^n$.*

Proof. Indeed, suppose that $g(\mathbf{y}) < \lfloor \bar{\eta}_x(\mathbf{y}) \rfloor$, which implies that $\bar{\eta}_x(\mathbf{y}) \geq 1$. By the property of CUA, there exist $k \equiv \lfloor \bar{\eta}_x(\mathbf{y}) \rfloor$ components of $\mathbf{y}$ whose average is at least x. Denote indices of these components by $I = \{i_1, \dots, i_k\}$. Consider vectors $\mathbf{y}_1, \dots, \mathbf{y}_k$ obtained from $\mathbf{y}$ by applying the cyclic permutation $(i_1, \dots, i_k)$ to its components i times for $\mathbf{y}_i$. Since g is symmetric, $g(\mathbf{y}_j) = g(\mathbf{y})$. Consider $\mathbf{y}' = \frac{1}{k}\sum_{j=1}^k \mathbf{y}_j$. Then $y'_i = \mathrm{UA}_k(\mathbf{y}) \geq x$ for all $i \in I$, and therefore $\eta_x(\mathbf{y}') \geq k$. Notice that since g is quasiconvex, $g(\mathbf{y}') \leq \max\{g(\mathbf{y}_1), \dots, g(\mathbf{y}_k)\} = g(\mathbf{y}) < k \leq \eta_x(\mathbf{y}')$. That is, this contradicts the assumption that g is greater than η_x. ∎

Notice that since function $\lfloor \cdot \rfloor$ is nondecreasing, and $\lfloor \max\{a, b\} \rfloor = \max\{\lfloor a \rfloor, \lfloor b \rfloor\}$, we have that $\lfloor \bar{\eta}_x \rfloor$ is itself a symmetric, quasiconvex function which is greater than η_x and that, moreover, it is the smallest function in the class of symmetric quasiconvex functions that are greater than η_x.

This may be quite useful in practice, particularly in an optimization context, as CUA_x can be viewed as an efficiently calculable upper bound for CUT_x.

3.2.2. Simultaneous calculation of CUA_x and CUT_x.

An important and interesting property of CUA_x calculation formula (4) is that calculation of CUA_x provides us with information about CUT_x. Specifically, we have the following property of (4).

Proposition 3. *Suppose for a vector $\mathbf{y} \in \mathbb{R}^n$ and a threshold $x \in (\mathrm{UA}_n(\mathbf{y}), \mathrm{UA}_1(\mathbf{y}))$ we have that*

$$\bar{\eta}_x(\mathbf{y}) = \sum_{i=1}^n [a^*(y_i - x) + 1]^+ = k \ ,$$

where a^* is an optimal point for (4). Then, if k is noninteger, we have that $a^* = \frac{1}{x - y^{(n - \lfloor k \rfloor)}}$ and is the unique minimizer of (4). Furthermore, we have that

$$\eta_{x - \frac{1}{a^*}}(\mathbf{y}) = \lceil k \rceil \ .$$

Otherwise, if k is integer, the set of minimizers of (4) is given by the interval $\mathcal{I} = [\frac{1}{x - y^{(n-k)}}, \frac{1}{x - y^{(n-k+1)}}]$, where $a^* \in \mathcal{I}$. Furthermore, we have that

$$\eta_{x - \frac{1}{a^*}}(\mathbf{y}) = \begin{cases} k + 1 & \text{if } a^* = \frac{1}{x - y^{(n-k)}} \ , \\ k & \text{otherwise} \ . \end{cases} \tag{16}$$

Proof. This result follows from a similar result in [12] and [9] regarding bPOE in a probabilistic setting. For the interested reader, a full proof in the CUA setting can be found in the extended version of this paper.[4] ∎

Proposition 3 is quite useful as it shows that simply calculating CUA_x provides information about CUT_x. Note that this result can be viewed as analogous to one regarding conditional-value-at-risk in [15], which shows that the value of value-at-risk can be calculated as a byproduct of the calculation formula (2).

In the context of our earlier example involving 10,000 servers, we said that $x \times \mathrm{CUA}_x$ told us how many additional resources were needed to cover the CUA_x busiest queues. Using Proposition 3, we can expand upon this statement with the following corollary.

[4]See www.ise.ufl.edu/uryasev/publications/.

COROLLARY 2. *Suppose that for a vector* $\mathbf{y} \in \mathbb{R}^n$ *and a threshold* $x \in (\mathrm{UA}_n(\mathbf{y}), \mathrm{UA}_1(\mathbf{y}))$ *we have that* $\bar{\eta}_x(\mathbf{y}) = \sum_{i=1}^n [a^*(y_i - x) + 1]^+ = k$. *Then*

$$x\bar{\eta}_x(\mathbf{y}) \le \sum_i \left\{ y_i \middle| y_i \ge x - \frac{1}{a^*} \right\} \le x(\lfloor \bar{\eta}_x(\mathbf{y}) \rfloor + 1) \ .$$

Proof. We first prove the left-hand inequality. Recall, as shown in the proof of Theorem 1, Case 1, that if

$$\bar{\eta}_x(\mathbf{y}) = \min_{x-\gamma>0} \frac{\sum_{i=1}^n [y_i - \gamma]^+}{x-\gamma} = \frac{\sum_{i=1}^n [y_i - \gamma^*]^+}{x - \gamma^*} = k \ , \tag{17}$$

then $\mathrm{UA}_k(\mathbf{y}) = x$ and $\gamma^* \in \mathrm{argmin}_\gamma\, \gamma + \frac{1}{k}\sum_{i=1}^n [y_i - \gamma]^+$. Rearranging and applying the change of variable $a^* = \frac{1}{x-\gamma^*}$, or equivalently $\gamma^* = x - \frac{1}{a^*}$, we have

$$\bar{\eta}_x(\mathbf{y})\left(x - \left(x - \frac{1}{a^*}\right)\right) = \sum_i \left[y_i - \left(x - \frac{1}{a^*}\right)\right]^+,$$

which then gives

$$x\bar{\eta}_x(\mathbf{y}) = \sum_i \left[y_i - \left(x - \frac{1}{a^*}\right)\right]^+ + \left(x - \frac{1}{a^*}\right)\bar{\eta}_x(\mathbf{y}) \ .$$

Noting that Proposition 3 implies that $\bar{\eta}_x(\mathbf{y}) \le \eta_{x-\frac{1}{a^*}}(\mathbf{y})$, we finally have that

$$\begin{aligned} x\bar{\eta}_x(\mathbf{y}) &= \sum_i \left[y_i - \left(x - \frac{1}{a^*}\right)\right]^+ + \left(x - \frac{1}{a^*}\right)\bar{\eta}_x(\mathbf{y}) \\ &\le \sum_i \left[y_i - \left(x - \frac{1}{a^*}\right)\right]^+ + \left(x - \frac{1}{a^*}\right)\eta_{x-\frac{1}{a^*}}(\mathbf{y}) = \sum_i \left\{ y_i \middle| y_i \ge x - \frac{1}{a^*} \right\} \quad \square \end{aligned}$$

Now we prove the right-hand inequality. First, note that if $\bar{\eta}_x(\mathbf{y}) = k$, then $\mathrm{UA}_k(\mathbf{y}) = x$. Second, note that Proposition 3 implies that $k \le \eta_{x-\frac{1}{a^*}}(\mathbf{y})$, implying further that $x = \mathrm{UA}_k(\mathbf{y}) \ge \mathrm{UA}_{\eta_{x-\frac{1}{a^*}}(\mathbf{y})}(\mathbf{y})$. Third, note that Proposition 3 also implies that $\eta_{x-\frac{1}{a^*}} \le \lfloor \bar{\eta}_x(\mathbf{y}) \rfloor + 1$. Finally, note that

$$\sum_i \left\{ y_i \middle| y_i \ge x - \frac{1}{a^*} \right\} = \mathrm{UA}_{\eta_{x-\frac{1}{a^*}}(\mathbf{y})}(\mathbf{y})\eta_{x-\frac{1}{a^*}} \ .$$

Using these facts, we have that

$$\sum_i \left\{ y_i \middle| y_i \ge x - \frac{1}{a^*} \right\} = \mathrm{UA}_{\eta_{x-\frac{1}{a^*}}(\mathbf{y})}(\mathbf{y})\eta_{x-\frac{1}{a^*}} \le x\eta_{x-\frac{1}{a^*}} \le x(\lfloor \bar{\eta}_x(\mathbf{y}) \rfloor + 1) \ .$$

Therefore, Corollary 2, in the context of the server example, shows us that the total amount of unserved jobs waiting in queues with length longer than or equal to $x - \frac{1}{a^*}$ is somewhere between $x\bar{\eta}_x(\mathbf{y})$ and $x(\lfloor \bar{\eta}_x(\mathbf{y}) \rfloor + 1)$.

4. Optimization of $\mathbf{CUA}_x$. When entered into an optimization setting, CUA_x provides substantial benefits, particularly when compared to the analogous use of CUT_x. Consider the following CUT_x minimization problem, where $S \subseteq \mathbb{R}^n$ is a convex set and $x \in \mathbb{R}$ is a specified threshold:

$$\min_{\mathbf{y}\in S} \eta_x(\mathbf{y}).$$

In order to solve this problem, it is natural to reformulate it as an MIP problem. In other words, to indicate the state of a vector component as being larger than x or not, introduction of binary variables is a natural consideration.

We consider the minimization of CUA_x, which is posed as follows:

$$\min_{\mathbf{y}\in S} \bar{\eta}_x(\mathbf{y}) = \min_{\mathbf{y}\in S} \left\{ \min_{a\geq 0} \sum_{i=1}^{n} [a(y_i - x) + 1]^+ \right\} = \min_{\mathbf{y}\in S, a\geq 0} \sum_{i=1}^{n} [a(y_i - x) + 1]^+ \ .$$

As we will show by multiple examples, if the set S is convex, the CUA_x minimization problem can be reduced to convex programming. Furthermore, if the set S is a polyhedron, the CUA_x minimization problem can be reduced to linear programming. This is, in fact, a general result which follows from [9]. Specifically, as we will show in specific cases, as long as S is convex, we can make a simple change of variable $ay_i = \hat{y}_i$ to yield the equivalent convex problems, i.e.,

$$\min_{\mathbf{y}\in S, a\geq 0} \sum_{i=1}^{n} [a(y_i - x) + 1]^+ = \min_{\hat{\mathbf{y}}\in \hat{S}, a\geq 0} \sum_{i=1}^{n} [\hat{y}_i - ax + 1]^+ \ ,$$

where $\hat{S} = \mathrm{cl}\ \mathrm{cone}(S)$ is a closed convex cone. Linear programming reformulations follow similarly.

Additionally, CUA_x minimization itself is meaningful, as it considers the magnitude of the components around the threshold x. Therefore, while also minimizing an upper bound for CUT_x, the CUA_x minimization is interesting in its own right. To demonstrate this, we provide example network flow problems where CUA_x optimization may be more appealing than the analogous CUT_x minimization.

4.1. Applications to network optimization. We now present variations of three standard network flow problems in the context that we have edges or nodes that become overloaded and we would thus like to minimize the number of edges/nodes that are overloaded. To solve network flow problems in this context, we use formulations with a CUA_x or CUT_x objective. We first show that the CUA_x minimization problems reduce to LP, yielding obvious computational advantages over the CUT_x minimization problems, which are formulated as MIP. We bolster this point by providing a computational example and by proving an NP-hardness result for CUT_x minimization problems. We then show that, depending on the context, the optimal CUT_x network flows can be unappealing, yielding edges or nodes that are severely overloaded or flow configurations that are more sensitive to minor fluctuations in network conditions. Optimal CUA_x flows, on the other hand, can be shown to mitigate these issues, providing relatively stable solutions with more evenly distributed flow patterns. The three network flow problems we discuss are the generalized assignment, min cost flow, and capacity planning problems.

Of course, both CUA_x and CUT_x have their merits, and we make no claim that one is always superior to the other. In general, MIP formulations have much more flexibility in terms of modeling capabilities as compared to LP formulations. Nevertheless, we focus on the benefits that the efficiently solvable CUA_x formulations can provide in comparison to CUT_x formulations.

4.1.1. The generalized assignment problem. Throughout the remaining sections, we consider a network represented by a graph $G = (V, E)$ of edges E and vertices V, where loads of product flow from supply vertices V_S, through the graph edges E and transshipment vertices V_T, to the destination demand vertices V_D. It is common in network flow problems to be concerned with the amount of flow going through each node, which we generally call the *load*. For example, consider the following generalized assignment problem from [2]. We must assign $|V_S|$ jobs to $|V_T|$ machines,[5] where A_{ij} is the unit cost of assigning job i to machine j, and t_{ij} is the time it takes machine j to process a unit of job i. Just as in [2], we assume f_{ij} is the amount of job i assigned to machine j and that we can assign partial jobs to machines (i.e., we do not require integral f_{ij} assignments). The goal, then, is to assign all jobs at minimal cost such that no machine is running for more than x time units:

$$
\begin{aligned}
\min_{f} \quad & \sum_{(i,j)\in E} f_{ij} A_{ij} \\
\text{s.t.} \quad & \sum_{(i,j)\in E} f_{ij} = 1, \forall i \in V_S \ , \\
& \sum_{(i,j)\in E} f_{ij} t_{ij} \le x, \forall j \in V_T \ , \\
& f_{ij} \ge 0 \ .
\end{aligned} \tag{18}
$$

For this problem, it is considered undesirable to have the time-*load* of a machine larger than x. A machine might, for example, become prone to malfunctions, overheating, or other risk factors if it becomes overloaded.

Consider, though, the case in which we would like to solve such a problem, but our x makes the problem infeasible, meaning that we must overload *some* of the machines. We could, of course, simply increase x, but that may cause us to overload a large number of machines. Thus, we would like to find an assignment that minimizes the number of machines that are overloaded. Also, to make sure our assignment has reasonable cost, we want it to satisfy a certain budget $\sum_{(i,j)\in E} f_{ij} A_{ij} \le B$. A natural way to formulate this problem is as a CUT_x minimization problem (19):

$$
\begin{aligned}
\min_{f,\xi,l} \quad & \sum_{j\in V_T} \xi_j \\
\text{s.t.} \quad & \xi_j \ge \frac{l_j - x}{M}, \forall j \in V_T \ , \\
& l_j = \sum_{(i,j)\in E} f_{ij} t_{ij}, \forall j \in V_T \ , \\
& \sum_{(i,j)\in E} f_{ij} = 1, \forall i \in V_S \ , \\
& \sum_{(i,j)\in E} f_{ij} A_{ij} \le B \ , \\
& f_{ij} \ge 0, \xi_{ij} \in \{0, 1\} \ .
\end{aligned} \tag{19}
$$

[5] In this context, we have no demand nodes and each supply node has supply equal to 1.

Here, ξ_j indicates whether the load l_j on machine j exceeds the threshold and M is a sufficiently large constant. Though this MIP formulation will solve the problem of minimizing the number of overloaded nodes (machines), it is ignoring potentially meaningful information about the magnitude of loads put upon nodes. Specifically, it does not account for the magnitude of loads put upon nodes in the overloaded state, as well as the nodes with loads less than, but most near to, the overloaded state. For nodes that are already overloaded, it may be important to consider by how much they are overloaded. Additionally, it may be undesirable to have nodes that, while not yet overloaded, are very close to being in the overloaded state. For example, assume that if a machine runs for too long, it may malfunction, and there is a large cost that must be paid to fix the broken machine. Let $I\{l_i > x\}$ be an indicator function, indicating if a node is overloaded. Now suppose that, for a given assignment configuration and associated loads l_i, the probability of a malfunction occurring at node i is given by

$$P(\text{malfunction at node } i) = 0.5I\{l_i > x\} + 0.5\left(1 - \frac{1}{e^{[l_i - x]^+}}\right) .$$

Therefore, we have that a transshipment node will not malfunction until it is overloaded and once overloaded, the probability jumps to 0.5 and increases exponentially based on the magnitude of the load vector. Solving the CUT_x minimization problem to calculate flow configuration for this system is ignoring critical information and may produce a very undesirable result. First, notice that the CUT_x minimization problem will ignore the magnitude with which a node is overloaded, thus ignoring the fact that *very* overloaded nodes will, with higher probability, incur the additional cost C. Second, notice that the CUT_x minimization will ignore information about nodes with loads less than, but very near to, x. This can cause major stability issues if there is uncertainty regarding exact system specifications at test-time. For example, consider a CUT_x minimization solution which has many nodes with loads less than, but very near to, x. If, at test-time, t_{ij} is slightly larger than expected, we will have a cascade of increasing malfunction probabilities as the nodes with load close to x are pushed over the threshold with their failure probability increasing from zero to around 0.5. In Example 1, we demonstrate numerically that CUT_x optimal solutions indeed exhibit these characteristics.

By minimizing CUA_x, we formulate a network flow problem that accounts for these factors (i.e., the magnitudes of loads). Specifically, we have the CUA_x optimization problem (20), which aims to minimize the number of most loaded nodes that have loads averaging to x:

$$\begin{aligned} \min_{y,z,a,l} \quad & \sum_{j\in V_T} z_j \ , \\ \text{s.t.} \quad & z_j \geq l_j - ax + 1 \ , \\ & l_j = \sum_{(i,j)\in E} y_{ij} t_{ij}, \forall j \in V_T \ , \\ & \sum_{(i,j)\in E} y_{ij} = a, \forall i \in V_S \ , \\ & \sum_{(i,j)\in E} y_{ij} A_{ij} \leq aB \ , \\ & a \geq 0, y_{ij} \geq 0, z_j \geq 0 \ . \end{aligned} \tag{20}$$

Table 1
Ordered list of optimal loads $l_j = \sum_{(i,j)\in E} f_{ij}t_{ij}$ for all $j \in V_T$. Overloaded nodes are bold face.

CUA	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.6	1.7	1.7
	2.0	2.2	2.2	2.3	2.5	**3.2**	**3.7**	**4.3**	**5.5**	**6.3**
CUT	0	2.2	2.4	2.5	2.5	2.5	2.5	2.5	2.5	2.5
	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	**92.7**

Notice that this has been reduced to LP via a simple change of variable $af_{ij} = y_{ij}$. This is a powerful result, which implies that we can achieve a similar goal (i.e., minimizing the number of nodes with loads exceeding or slightly below a specified threshold) using LP as opposed to MIP. In terms of the network optimization problem, it considers how much load is burdening the overloaded nodes (i.e., by how much they are overloaded). Additionally, it considers how much load is burdening the nodes with loads less than, but most near to, the threshold (i.e., how close to being overloaded are the most burdened nodes). Consider the following numerical example.

Example 1. We consider this problem with 50 jobs and 20 machines, where $(i, j) \in E$ with probability 0.5, A_{ij} are uniformly distributed on $[0, 10]$, and t_{ij} are uniformly distributed on $[0, 1]$. For our randomly generated task, we find that the general assignment problem is infeasible at $x = 2.5$, so we consider this our threshold for CUA_x and CUT_x minimization. We also set $B = 25$, which was set such that the budget constraint would be tight for both CUA_x and CUT_x optimal assignments (i.e., to make sure we had reasonable assignments). In Table 1 we show the ordered list of loads given by the optimal CUA_x and CUT_x assignments. We see that CUT_x severely overloads a single node in order to drive CUT_x to 1. The CUA_x solution, on the other hand, is spread much more evenly. For example, we see that the sum of nodes exceeding x equals 23 for CUA_x and 92.7 for CUT_x. We also see that the CUT_x solution drives many loads to equal the threshold. As already mentioned, this can lead to stability issues, as uncertainty in t_{ij} could lead to many machines being pushed over the time limit threshold. If there is a discrete jump in failure probability as a load surpasses x, this could lead to heightened probability for a cascade of failures or a huge jump in the expected number of machine malfunctions. Note also that we could have introduced upper bounds $l_j \leq U_j$ on the loads in the CUT_x formulation to prevent this extreme solution. However, the CUT_x minimization will still have the same properties, overloading a few machines to the maximum, U_j, to drive down CUT_x with many other machines having $l_j = x$. Additionally, it is still an MIP solution compared to a potential LP solution.

Remark. Other objectives could be used for the optimization problems considered in this section that also consider the magnitudes of the largest components and are convex. One could use UA_k or the overall exceedance $\sum_i [l_i - x]^+$. However, these have drawbacks compared to CUA_x. For UA_k, although it is actually equivalent to CUA_x for the proper choice of k, one will still need to choose an appropriate k. In our examples, a threshold is natural, but choosing the correct k is less straightforward. It is hard to tell how many of the largest components you want to minimize. For overall exceedance, there are two major differences. First, it does not count components. Thus, the expected exceedance may be small, but there may be a large number of components comprising this sum. Second, it does not consider the magnitude of the largest components that are *less* than the threshold. As already mentioned, this may be important to consider. In a separate set of experiments, we saw that minimization

TABLE 2

Number of transshipment nodes, T, and approximate time to solve with Gurobi in Python interface.

T	MIP run time (seconds)	LP run time (seconds)
100	2244.94	31.53
75	492.42	14.52
50	191.34	13.23
25	109.33	5.95
10	21.26	6.47

of overall exceedance often gives you many components lying on the threshold with $l_i = x$.

Example 2. For a simple computational demonstration comparing the performance of the CUA_x formulation versus the CUT_x formulation, we consider a formulation similar to the generalized assignment problem, with the only change being an extra demand node which is connected to all $i \in V_T$ with cost of transmission being t_{id} for all $i \in V_T$. One possible interpretation comes from an information flow networks area, just with the flow going in the opposite direction. The demand vertex is associated with a data server, supply vertices are associated with end users, and transshipment nodes are associated with routing servers.[6] The goal is to assign routing in a such way that the routing servers are not overloaded with information flow. The CUA_x and CUT_x formulations would be identical to (20) and (19) but with the loads including an extra term to equal $f_{jd}t_{jd} + \sum_{(i,j)\in E} f_{ij}t_{ij}$ for all $j \in V_T$ and flow conservation constraints for transshipment and demand nodes.

Using this network structure, we solved different-sized problems using Gurobi Solver on a PC via a Python interface. The Python code and full problem description can be found online.[7] Note that results are reported for random graph instances having nontrivial solutions where the optimal objective does not equal 0 or $|V_T|$: $|V_S| = 10{,}000$; $|V_T| = T$; $|V_D| = 1$; supply per supply node $= 10$; demand for demand node $= 100{,}000 = (10) \times (10{,}000)$; t_{ij} uniformly distributed in $[0, 1]$; threshold for $\text{CUA}_x = x = 0.5 \times 100{,}000/T$; threshold for $\text{CUT}_x = x = 0.2 \times 100{,}000/T$; $M = 100{,}000$.

With this setup, we compared the performance of the MIP CUT_x minimization formulation and the LP CUA_x minimization formulation for varying values of T. Run time comparisons can be seen in Table 2. We see that the LP formulation has a clear advantage as the number of transshipment nodes, i.e., the number of binary variables in the MIP formulation, increases. Thus, we see that the CUA_x minimization problem is significantly faster. For large problems, CUA_x solving time will be dramatically lower than MIP solving time.

4.1.2. Min cost network flows. While the first two examples considered the load put upon nodes, it is also quite natural to consider the load (or flow) on individual arcs. The next two examples discuss problems of this type, where we would like to minimize the number of arcs with large flow or large overall flow costs. One of the

[6]In this interpretation, flow goes from demand to supply but the optimal flows are equivalent.

[7]See www.ise.ufl.edu/uryasev/research/testproblems/logistics/network-optimization-by-minimization-of-cardinality-of-upper-averages-cua/.

most common network flow formulations is the min cost network flow problem [1]:

$$\begin{aligned} \min_{f} \quad & \sum_{(i,j)\in E} c_{ij} f_{ij} \\ \text{s.t.} \quad & \sum_{(i,j)\in E} f_{ij} - \sum_{(k,i)\in E} f_{ki} = b_i \ \forall i \in V \ , \\ & 0 \le f_{ij} \le u_{ij} \ . \end{aligned} \tag{21}$$

In this formulation, f_{ij} indicates the flow through edge $(i,j) \in E$, c_{ij} is the unit cost of flow, u_{ij} is the capacity, b_i is the demand at vertex $i \in V$, where $b_i = 0$ for $i \in V_T$, $b_i > 0$ for $i \in V_D$, and $b_i < 0$ for $i \in V_S$. However, this problem makes the critical assumption that all edge costs $c_{ij} f_{ij}$ are equally important, which may not be a valid assumption (see, e.g., [1, Chapter 14]). It might be the case that we would not want any edge cost to exceed some level x. For example, x may represent some budget associated with each arc and we do not want to overload too many budgets. Of course, this may not be possible for all edges and we may be forced to overload some budgets. Thus, this leads us to two possibilities for a risk averse min cost network flow problem where we would like to minimize the number of edge costs that are large w.r.t. some threshold x. We have the CUA_x minimization (22), which has been reduced to an LP formulation by the change of variable $af_{ij} = y_{ij}$, and the CUT_x minimization (23), which has been reduced to an MIP formulation:

$$\begin{aligned} \min_{y,z,a} \quad & \sum_{(i,j)\in E} z_{ij} \\ \text{s.t.} \quad & z_{ij} \ge c_{ij} y_{ij} - ax + 1 \ , \\ & \sum_{(i,j)\in E} y_{ij} - \sum_{(k,i)\in E} y_{ki} = ab_i \ , \\ & 0 \le y_{ij} \le au_{ij} \ , \\ & a \ge 0, z_{ij} \ge 0 \ , \end{aligned} \tag{22}$$

$$\begin{aligned} \min_{f,\xi} \quad & \sum_{(i,j)\in E} \xi_{ij} \\ \text{s.t.} \quad & \sum_{(i,j)\in E} f_{ij} - \sum_{(k,i)\in E} f_{ki} = b_i \ , \\ & 0 \le f_{ij} \le u_{ij} \ , \\ & \xi_{ij} \ge \frac{c_{ij} f_{ij} - x}{M} \ , \\ & \xi_{ij} \in \{0,1\} \ . \end{aligned} \tag{23}$$

Example 3. Consider the network in Figure 3, where we would like to push 24 units of supply through the network from the left black node to the right black node. We have edge costs uniformly distributed on $[0,1]$ and we assume that each arc has unbounded capacity and we set $x = 3$. The optimal CUA_x, CUT_x, and min cost flows are shown in Figure 3. We notice, first, that the min cost flow has only 4 edges with cost exceeding 3, but these costs are fairly large. The CUA_x solution, on the other hand, has three edges with cost exceeding 3, but with much smaller magnitude. The CUT_x solution only has one edge exceeding 3, but the magnitude of this edge is more

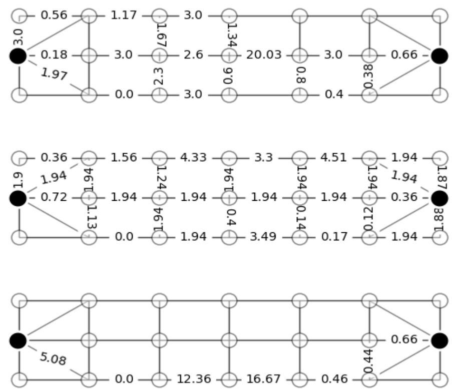

FIG. 3. *(Top) Optimal CUT flow. (Middle) Optimal* CUA_x *flow. (Bottom) Min cost flow. Numbers on arcs represent* $f_{ij}c_{ij}$.

than four times the magnitude of the largest CUA_x edge cost. We also see that the CUA_x flow is more spread out, with more edges being utilized. In addition, we see that the utilization is directly influenced by the threshold. With the CUT_x flow, we have some edge costs equal to the threshold, while the CUA_x flow is almost opposite in that the largest edge cost that is less than or equal to 3 is equal to 1.93. Thus, there is a buffer between these smaller edge costs and the threshold, which may be useful if costs or other parameters are uncertain.

4.1.3. CUA and capacity planning. The capacity planning network flow problem considers the situation where the current network is composed of arcs with capacity that is not large enough that all demand can be satisfied. Therefore, the task is to figure out which arcs should have their capacity increased so that demand can be satisfied. Of course, many variants of this problem exist that are tailored to specific applications [11, 5, 4]. We consider, though, a generic variant where each arc has initial capacity x, and we assume it will cost C to expand the capacity of any arc to u.[8] This problem can be posed as (24), a CUT_x minimization network flow problem where we expand capacity of the minimal number of arcs so that demand can be satisfied:

$$
\begin{aligned}
\min_{f,\xi} \quad & \sum_{(i,j)\in E} \xi_{ij} \\
\text{s.t.} \quad & \xi_{ij} \geq \frac{f_{ij}-x}{M}\,, \\
& \sum_{(i,j)\in E} f_{ij} - \sum_{(k,i)\in E} f_{ki} = b_i\,, \\
& 0 \leq f_{ij} \leq u\,, \\
& \xi_{ij} \in \{0,1\}\,.
\end{aligned} \tag{24}
$$

However, as we show in Example 4, the CUT_x optimal solution can have unappealing properties. Specifically, for the case of capacity planning, we see that the CUT_x optimal flow will likely fill all arcs to capacity (either u or x). This means that the

[8]Arc expansion is a binary decision with fixed cost for all arcs.

network may be very sensitive to changes in demand or arc failures. For example, if one arc fails, it is unlikely that flow will be able to be diverted because all other arcs are filled to capacity. As discussed in [11, 4], networks often require some capacity buffer so that uncertainties do not lead to massive disruptions. Power grids are a specific example considered in [4], where capacities on power lines are often violated to serve all demand when lines may fail or some unplanned flow must be transmitted across a power line. Thus, power grids, if designed without a buffer, can become susceptible to cascading failure, or blackouts.

We consider the similar problem of minimizing CUA_x, formulated as (25), where we have again reduced to LP via the change of variable $y_{ij} = af_{ij}$:

$$\begin{aligned} \min_{y,z,a} \quad & \sum_{(i,j)\in E} z_{ij} \\ \text{s.t.} \quad & z_{ij} \geq y_{ij} - ax + 1 \ , \\ & \sum_{(i,j)\in E} y_{ij} - \sum_{(k,i)\in E} y_{ki} = ab_i \ , \\ & 0 \leq y_{ij} \leq au \ , \\ & a \geq 0, z_{ij} \geq 0 \ . \end{aligned} \tag{25}$$

As we show in Example 4, the CUA_x optimal flow does not suffer from the same drawbacks as the CUT_x optimal flow. By considering the magnitude of components around the threshold x, the CUA_x optimal flow is more evenly spread among the arcs.

Of course, there are methods to directly mitigate the risk of cascading failures by introducing new elements into the CUT_x minimization formulation that consider uncertainty. For example, [4] considers a scenario-based approach, where flows must satisfy demand for multiple scenarios, while [13] implements a robust optimization approach in a similar fashion. However, introduction of additional components into an MIP formulation makes a numerically difficult optimization problem even more challenging. On the other hand, without having to gather scenario information or consider uncertainty directly, the CUA_x optimal flow is able to mitigate some of the risks associated with the CUT_x optimal flow and may be less susceptible to cascading failures. In addition, it is a simple LP formulation, and uncertainty considerations can still be entered into the formulation just as is done with the CUT_x formulations.

Example 4. Consider the network in Figure 4, where we would like to push 24 units of supply through the network from the left black node to the right black node. Assume that each arc has initial capacity $x = 7$, which is not sufficient to push flow through the network, so we must decide which arcs to expand to capacity $u = 10$. We assume that each expansion is a discrete decision that comes with a fixed cost C. We see CUA_x and CUT_x optimal flows in Figure 4. First, we can see that if the CUT_x optimal flow utilizes an arc, it likely fills it to capacity (either 7 or 10). In contrast, we notice that the CUA_x optimal flow is more evenly distributed throughout the network with all arcs having some buffer between their actual flow and their capacity (7 or 10). We can measure the susceptibility to arc failure by measuring the expected amount of lost flow given a single arc failure, assuming equally probable failures,[9] we find that

[9]To solve for the lost flow given an arc failure, we first take the network with the single arc removed where all other arcs have capacity x or u (depending on whether they were expanded or not). We then solve a max flow problem for that network where the output of the supply node is limited to the original demand, which in this example equals 24. The lost flow from dropping the single arc is then 24 minus the max flow for this new network.

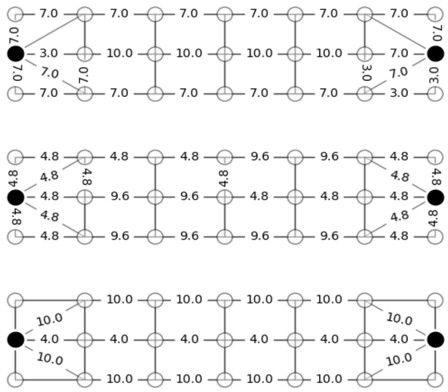

FIG. 4. *(Top) Optimal CUT flow. (Middle) Optimal* CUA_x *flow. (Bottom) Min cost flow with arc costs equal to* 1 *and arc upper bound equal to* 10*. Numbers on arcs represent* f_{ij}*.*

the CUA_x flow has expected loss equal to 2 and that the CUT_x flow has expected loss equal to 2.66.

4.1.4. $\mathbf{CUT}_x$ minimization is NP hard. It should be noted that the MIP formulations we consider for CUT_x minimization problems are big-M-type formulations. Though there may be more efficient methods for formulating and approximating such an MIP formulation, e.g., by considering a disjunctive programming approach, we emphasize that the CUT_x minimization problem is NP hard. In this section, we prove that for arbitrary graphs the CUT_x minimization problem is NP hard. Therefore, regardless of the strategy used to solve (exactly or approximately) the CUT_x minimization problem, CUA_x minimization will reduce to convex and linear programming, which have polynomial-time solvers.

In Proposition 4 we show that the problem of minimizing CUT_x for an arbitrary graph is an NP-hard problem with polynomial-time reduction from an NP-complete set covering problem to a CUT_x minimization problem. The NP-hardness implies that the considered problem is at least as hard as any P, NP, or NP-complete problem, that is, time consumed by solving this problem will probably grow exponentially with the size of the problem.

PROPOSITION 4. CUT_x *minimization is an NP-hard problem.*

Proof. The set covering problem is NP-complete. Solving a set-covering problem with one run of the CUT_x minimization problem of the same size will prove that CUT_x minimization problem is NP-hard. Suppose there is a set S consisting of sets S_i: $S = \{S_1, \dots, S_n\}$, where $S_i \subseteq U \equiv \{1, \dots, m\} = \bigcup_{i=1}^{n} S_i$. It is required to find a minimal covering set $S^* = \{S_1^*, \dots, S_k^*\}$ with $S_k^* \in S$ and minimal k. Consider a three-layer graph. On the first layer of the graph there are m demand vertices with demand 1, each vertex corresponds to an object from the union set U. On the second layer of the graph there are n transshipment nodes with demand/supply value 0, which correspond to sets S_i. The first-layer vertex j and the second-layer vertex i are connected iff $j \in S_i$. The transportation costs along all edges are equal to 1. The third layer of the graph consists of a single auxiliary supply vertex with the supply m. CUT_x is measured on the vector of loads for transshipment vertices which are the same as those presented in Example 2, and the critical threshold is 0. That is, if a transshipment vertex is involved in transportation, its load is larger than 0 and it violates the threshold. Finally, for the described graph it can be seen that picking a covering subset with a minimal number of sets is equivalent to minimizing the number of overloaded transshipment vertices. □

5. Conclusion. In this paper, we have introduced a new concept called CUA_x, a function which counts the number of largest outcomes in a data set which have average value equal to a specified threshold, x. As a basic characteristic for counting tail outcomes, CUA_x not only counts outcomes like CUT_x, but accounts for the severity of these outcomes, which CUT_x does not. We have also shown that CUA_x has superior mathematical properties, being a continuous function w.r.t. the threshold parameter, piecewise linear in its reciprocal, and directly optimizable via convex and linear programming. Thus, CUA_x can be efficiently calculated, has continuity properties which can be used for sensitivity analysis, and can be used to efficiently minimize the number of outcomes in the tail of the data set, with the tail including the largest outcomes that average to a specified threshold. We have shown that CUA_x is the inverse of UA_k, a function that measures the average magnitude of the largest k outcomes in a data set. We have also proved that CUA_x and CUT_x are strongly connected, showing that CUA_x is, in a certain sense, the minimal quasiconvex upper bound of CUT_x.

Finally, we have shown that CUA_x can be used to formulate new network optimization problems. We compare against similar formulations which minimize CUT_x. In addition to these formulations being less efficiently solvable, since they involved binary variables, we find that the solution can often be unappealing. We show that CUA_x optimization, on the other hand, reduces to solving a LP problem and that accounting for the severity of the largest outcomes can lead to more appealing network flow policies.

Appendix A. $\mathbf{CUA}_x$: A special case of bPOE.

A.1. bPOE and tail probabilities. When working with optimization of tail probabilities, one frequently works with constraints or objectives involving *Probability of Exceedance* (POE), $p_x(X) = P(X > x)$, or its associated quantile $q_\alpha(X) = \min\{x|P(X \leq x) \geq \alpha\}$, where $\alpha \in [0,1]$ is a probability level. The quantile is a popular measure of tail probabilities in financial engineering, called within this field value-at-risk due to its interpretation as a measure of tail risk. The quantile, though, when included in optimization problems via constraints or objectives, is quite difficult to treat with continuous (linear or nonlinear) optimization techniques.

A significant advancement was made in [15] in the development of an approach to combat the difficulties raised by the use of the quantile function in optimization. Rockafellar and Uryasev explored a replacement for the quantile, called CVaR within the financial literature, and called the superquantile in a general context. The superquantile is a measure of uncertainty similar to the quantile, but with superior mathematical properties. Formally, the superquantile (CVaR) for a continuously distributed X is expressed as

$$\bar{q}_\alpha(X) = E\left[X|X > q_\alpha(X)\right] .$$

For general distributions, the superquantile is defined by the formula

$$\bar{q}_\alpha(X) = \min_\gamma \left\{\gamma + \frac{E[X-\gamma]^+}{1-\alpha}\right\} ,$$

where $[\cdot]^+ = \max\{\cdot, 0\}$. Similar to $q_\alpha(X)$, the superquantile can be used to assess the tail of the distribution. The superquantile, though, is far easier to handle in optimization contexts. It also has the important property that it considers the magnitude of events within the tail. Therefore, in situations where a distribution may have a heavy tail, the superquantile accounts for magnitudes of low-probability large-loss tail events, while the quantile does not account for this information.

Working to extend this concept, bPOE was developed as the inverse of the superquantile in the same way that POE is the inverse of the quantile. Specifically, there exists two slightly different variants of bPOE, namely lower and upper bPOE. Paper [9] defines so-called lower bPOE in the following way, where $\sup X$ denotes the essential supremum of the random variable X.

Definition (lower bPOE). *Let X be a real-valued random variable and $x \in \mathbb{R}$ be a fixed threshold parameter. Lower bPOE of random variable X at threshold x equals*

$$\bar{p}^L_x(X) = \begin{cases} 0 & \text{if } x \geq \sup X \ , \\ \{1-\alpha | \bar{q}_\alpha(X) = x\} & \text{if } E[X] < x < \sup X \ , \\ 1 & \text{otherwise} \ . \end{cases}$$

In words, for any threshold $x \in (E[X], \sup X)$, lower bPOE can be interpreted as one minus the probability level at which the superquantile equals x.

Similarly, [12] defines so-called upper bPOE as follows.

Definition (upper bPOE). *Upper bPOE of random variable X at threshold x equals*

$$\bar{p}^U_x(X) = \begin{cases} \max\{1-\alpha | \bar{q}_\alpha(X) \geq x\} & \text{if } x \leq \sup X \ , \\ 0 & \text{otherwise} \ . \end{cases}$$

Upper and lower, in fact, do not differ dramatically. This is shown by the following property, proved in [12].

Upper vs. lower bPOE.

$$\bar{p}^U_x(X) = \begin{cases} \bar{p}^L_x(X) & \text{if } x \neq \sup X \ , \\ P(X = \sup X) & \text{if } x = \sup X \ . \end{cases}$$

It is important to notice that upper and lower bPOE are equivalent when $x \neq \sup X$. The difference between the two definitions arises when the threshold $x = \sup X$. In this case, we have that $\bar{p}^L_x(X) = 0$ while $\bar{p}^U_x(X) = P(X = \sup X)$. Thus, for a threshold $x \in (E[X], \sup X)$, both upper and lower bPOE of X at x can be interpreted as one minus the probability level at which the superquantile equals x. Roughly speaking, upper bPOE can be compared with $P(X \geq x)$ while lower bPOE can be compared with $P(X > x)$. To read further about the differences between upper and lower bPOE, see [9].

A.2. $\mathbf{CUA}_x$ and upper bPOE. In [12], an important calculation formula for bPOE was introduced. Specifically, [12] found that upper bPOE has the following calculation formula.

PROPOSITION. *Given a real-valued random variable X and a fixed threshold x, bPOE for random variable X at x equals*

$$\bar{p}_x^U(X) = \inf_{\gamma < x} \frac{E[X-\gamma]^+}{x-\gamma}$$
$$= \begin{cases} \lim_{\gamma \to -\infty} \frac{E[X-\gamma]^+}{x-\gamma} = 1 & \text{if } x \le E[X] \ , \\ \min_{\gamma < x} \frac{E[X-\gamma]^+}{x-\gamma} & \text{if } E[X] < x < \sup X \ , \\ \lim_{\gamma \to x^-} \frac{E[X-\gamma]^+}{x-\gamma} = P(X = \sup X) & \text{if } x = \sup X \ , \\ \min_{\gamma < x} \frac{E[X-\gamma]^+}{x-\gamma} = 0 & \text{if } \sup X < x \ . \end{cases} \tag{26}$$

With this calculation formula, we can then show that CUA_x is simply a special case of upper bPOE. First, let us represent our deterministic vector $(y_1, y_2, \dots, y_n) = \mathbf{y} \in \mathbb{R}^n$ as a real-valued discrete random variable Y taking on values $(y_1, y_2, \dots, y_n)$ with equal probabilities, i.e., $P(Y = y_i) = \frac{1}{n}$. Second, let us consider the quantity $n\bar{p}_x(Y)$. Using calculation formula (26) with the change of variable $a = \frac{1}{x-\gamma}$, we see that

$$n\bar{p}_x^U(Y) = \min_{a \ge 0} nE[a(Y-x)+1]^+$$
$$= \min_{a \ge 0} \sum_{i=1}^{n} [a(y_i - x) + 1]^+ \ . \tag{27}$$

Thus, we see that this is exactly the definition of CUA_x. In other words, we see that CUA_x is a deterministic variant of bPOE.

Acknowledgments. The authors would like to thank Dr. Vladimir Boginski, Prof. Jean-Philippe P. Richard, and Prof. R. Tyrrell Rockafellar for productive discussions and valuable comments.

REFERENCES

[1] R. K. AHUJA, T. L. MAGNANTI, AND J. B. ORLIN, *Network Flows: Theory, Algorithms, and Applications*, 1st ed., Pearson, London, 1993.
[2] D. P. BERTSEKAS, *Network Optimization: Continuous and Discrete Models*, Athena Scientific, Belmont, MA, 1998.
[3] D. BIENSTOCK, M. CHERTKOV, AND S. HARNETT, *Chance-constrained optimal power flow: Risk-aware network control under uncertainty*, SIAM Rev., 56 (2014), pp. 461–495.
[4] D. BIENSTOCK AND S. MATTIA, *Using mixed-integer programming to solve power grid blackout problems*, Discrete Optim., 4 (2007), pp. 115–141.
[5] S. BINATO, M. V. F. PEREIRA, AND S. GRANVILLE, *A new Benders decomposition approach to solve power transmission network design problems*, IEEE Trans. Power Syst., 16 (2001), pp. 235–240.
[6] D. BERTSIMAS, D. PACHAMANOVA, AND M. SIM, *Robust linear optimization under general norms*, Oper. Res. Lett., 32 (2004), pp. 510–516.
[7] J. R. DAVIS AND S. URYASEV, *Analysis of tropical storm damage using buffered probability of exceedance*, Nat. Hazards, 83 (2016), 465–483.
[8] H. FÖLLMER AND A. SCHIED, *Stochastic Finance: An Introduction in Discrete Time*, de Gruyter Textbook, 3rd ed., de Gruyter, Berlin, 2011.
[9] A. MAFUSALOV AND S. URYASEV, *Buffered Probability of Exceedance: Mathematical Properties and Optimization Algorithms*, Research report 2014-1, University of Florida, Gainesville, FL, 2015.
[10] A. MAFUSALOV AND S. URYASEV, *Conditional Value-at-Risk (CVaR) Norm: Stochastic Case*, Research report 2013-5, University of Florida, Gainesville, FL, 2014.

[11] T. L. Magnanti and R. T. Wong, *Network design and transportation planning: Models and algorithms*, Transp. Sci., 18 (1984), pp. 1–55.

[12] M. Norton and S. Uryasev, *Maximization of AUC and Buffered AUC in Binary Classification*, Research report 2014-2, University of Florida, Gainesville, FL, 2014.

[13] F. Ordóñez and J. Zhao, *Robust capacity expansion of network flows*, Networks, 50 (2007), pp. 136–145.

[14] K. Pavlikov and S. Uryasev, *CVaR norm and applications in optimization*, Optim. Lett., 8 (2014), pp. 1999–2020.

[15] R. T. Rockafellar and S. Uryasev, *Optimization of conditional value-at-risk*, J. Risk, 2 (2000), pp. 21–41.

[16] R. T. Rockafellar, *Safeguarding strategies in risky optimization*, Presentation, International Workshop on Engineering Risk Control and Optimization, University of Florida, Gainesville, FL, 2009.

[17] R. T. Rockafellar and J. O. Royset, *On buffered failure probability in design and optimization of structures*, Reliab. Engin. Syst. Safety, 95 (2010), pp. 499–510.

[18] S. Uryasev, *Buffered Probability of Exceedance and Buffered Service Level: Definitions and Properties*, Research report 2014-3, University of Florida, Gainesville, FL, 2014.

[19] R. R. Yager, *On ordered weighted averaging aggregation operators in multicriteria decision making*, IEEE Trans. Syst. Man Cybernet., 18 (1988), pp. 183–190.