Surveys in Operations Research and Management Science 18 (2013) 1–17

Contents lists available at ScienceDirect

Surveys in Operations Research and Management Science

journal homepage: www.elsevier.com/locate/sorms

Review

Practical guidelines for solving difficult linear programs

Ed Klotz [a], Alexandra M. Newman [b,*]

[a] IBM, 926 Incline Way, Suite 100, Incline Village, NV 89451, United States

[b] Division of Economics and Business, Colorado School of Mines, Golden, CO 80401, United States

ARTICLE INFO

Article history:
Received 26 July 2012
Received in revised form
4 November 2012
Accepted 7 November 2012

ABSTRACT

The advances in state-of-the-art hardware and software have enabled the inexpensive, efficient solution of many large-scale linear programs previously considered intractable. However, a significant number of large linear programs can require hours, or even days, of run time and are not guaranteed to yield an optimal (or near-optimal) solution. In this paper, we present suggestions for diagnosing and removing performance problems in state-of-the-art linear programming solvers, and guidelines for careful model formulation, both of which can vastly improve performance.

Contents

1. Introduction

Operations research practitioners have been formulating and solving linear programs since the 1940s [1]. State-of-the-art optimizers such as CPLEX [2], Gurobi [3], MOPS [4], Mosek [5], and Xpress-MP [6] can solve most practical large-scale linear programs effectively. However, some "real-world" problem instances require days or weeks of solution time. Furthermore, improvements in computing power and linear programming solvers have often prompted practitioners to create larger, more difficult linear programs that provide a more accurate representation of the systems they model. Although not a guarantee of tractability, careful model formulation and standard linear programming algorithmic tuning often result in significantly faster solution times, in some cases admitting a feasible or near-optimal solution which could otherwise elude the practitioner.

In this paper, we briefly introduce linear programs and their corresponding commonly used algorithms, show how to assess optimizer performance on such problems through the respective algorithmic output, and demonstrate methods for improving that performance through careful formulation and algorithmic parameter tuning. We assume basic familiarity with fundamental mathematics, such as matrix algebra, and with optimization. We expect that the reader has formulated linear programs and has a conceptual understanding of how the corresponding problems can be solved. The interested reader can refer to basic texts such as [7–11] for more detailed introductions to mathematical programming, including geometric interpretations. For a more general discussion of good optimization modeling practices, we refer the reader to [12]. However, we contend that this paper is self-contained such that relatively inexperienced practitioners can use its advice effectively without referring to other sources.

* Corresponding author. Tel.: +1 303 273 3688; fax: +1 303 273 3416.
E-mail addresses: klotz@us.ibm.com (E. Klotz), newman@mines.edu (A.M. Newman).

1876-7354/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.sorms.2012.11.001

The remainder of the paper is organized as follows. In Section 2, we introduce linear programs and the simplex and interior point algorithms. We also contrast the performance of these algorithms. In Section 3, we address potential difficulties when solving a linear program, including identifying performance issues from the corresponding algorithmic output, and provide suggestions to avoid these difficulties. Section 4 concludes the paper with a summary. Sections 2.1 and 2.2, with the exception of the tables, may be omitted without loss of continuity for the practitioner interested only in formulation and algorithmic parameter tuning without detailed descriptions of the algorithms themselves. To illustrate the concepts we present in this paper, we show output logs resulting from having run a state-of-the-art optimizer on a standard desktop machine. Unless otherwise noted, this optimizer is CPLEX 12.2.0.2, and the machine possesses four 3.0 GHz Xeon chips and eight gigabytes of memory.

2. Linear programming fundamentals

We consider the following system where x is an $n \times 1$ column vector of continuous-valued, nonnegative decision variables, A is an $m \times n$ matrix of left-hand-side constraint coefficients, c is an $n \times 1$ column vector of objective function coefficients, and b is an $m \times 1$ column vector of right-hand-side data values for each constraint.

$$(P_{\mathrm{LP}}): \min \quad c^T x$$
$$\text{subject to } Ax = b$$
$$x \geq 0.$$

Though other formats exist, without loss of generality, any linear program can be written in the primal *standard form* we adopt above. Specifically, a maximization function can be changed to a minimization function by negating the objective function (and then negating the resulting optimal objective). A less-than-or-equal-to or greater-than-or-equal-to constraint can be converted to an equality by adding a nonnegative *slack variable* or subtracting a nonnegative *excess variable*, respectively. Variables that are unrestricted in sign can be converted to nonnegative variables by replacing each with the difference of two nonnegative variables.

We also consider the related *dual problem* corresponding to our primal problem in standard form, (P_{LP}). Let y be an $m \times 1$ column vector of continuous-valued, unrestricted-in-sign decision variables; A, b and c are data values with the dimensionality given above in (P_{LP}).

$$(D_{\mathrm{LP}}): \max \; y^T b$$
$$\text{subject to } y^T A \leq c^T.$$

The size of a linear program is given by the number of constraints, m, the number of variables, n, and the number of non-zero elements in the A matrix. While a large number of variables affects the speed with which a linear program is solved, the commonly used LP algorithms solve linear systems of equations dimensioned by the number of constraints. Because these linear solves often dominate the time per iteration, the number of constraints is a more significant measure of solution time than the number of variables. Models corresponding to practical applications typically contain sparse A matrices in which more than 99% of the entries are zero. In the context of linear programming, a dense matrix need not have a majority of its entries assume non-zero values. Instead, a dense LP matrix merely has a sufficient number or pattern of non-zeros so that the algorithmic computations using sparse matrix technology can be sufficiently time consuming to create performance problems. Thus, a matrix can still have fewer than 1% of its values be non-zero, yet be considered dense. Practical examples of such matrices include those that average more than 10 non-zeros per column and those with a small subset of columns with hundreds of non-zeros. State-of-the-art optimizers capitalize on matrix sparsity by storing only the non-zero matrix coefficients. However, even for such matrices, the positions of the non-zero entries and, therefore, the ease with which certain algorithmic computations (discussed below) are executed, can dramatically affect solution time.

A *basis* for the primal system consists of m variables whose associated matrix columns are linearly independent. The basic variable values are obtained by solving the system $Ax = b$ given that the resulting $n-m$ non-basic variables are set to values of zero. The set of the actual values of the variables in the basis, as well as those set equal to zero, is referred to as a *basic solution*. Each primal basis uniquely defines a basis for the dual (see [1, pp. 241–242]). For each basis, there is a range of right-hand-side values such that the basis retains the same variables. Within this range, each constraint has a corresponding dual variable value which indicates the change in the objective function value per unit change in the corresponding right hand side. (This variable value can be obtained indirectly via the algorithm and is readily available through any standard optimizer.) Solving the primal system, (P_{LP}), provides not only the primal variable values but also the dual values at optimality; these values correspond to the optimal variable values to the problem (D_{LP}). Correspondingly, solving the dual problem to optimality provides both the optimal dual and primal variable values. In other words, we can obtain the same information by solving either (P_{LP}) or (D_{LP}).

Each vertex (or extreme point) of the polyhedron formed by the constraint set $Ax = b$ corresponds to a basic solution. If the solution also satisfies the nonnegativity requirements on all the variables, it is said to be a *basic feasible solution*. Each such basic feasible solution, of which there is a finite number, is a candidate for *an* optimal solution. In the case of multiple optima, any convex combination of extreme-point optimal solutions is also optimal. Because basic solutions contain significantly more zero-valued variables than solutions that are not basic, practitioners may be more easily able to implement basic solutions. On the other hand, basic solutions lack the "diversity" in non-zero values that solutions that are not basic provide. In linear programs with multiple optima, solutions that are not basic may be more appealing in applications in which it is desirable to spread out the non-zero values among many variables. Linear programming algorithms can operate with a view to seeking basic feasible solutions for either the primal or for the dual system, or by examining solutions that are not basic.

2.1. Simplex methods

The practitioner familiar with linear programming algorithms may wish to omit this and the following subsection. The primal simplex method [1], whose mathematical details we provide later in this section, exploits the linearity of the objective and convexity of the feasible region in (P_{LP}) to efficiently move along a sequence of extreme points until an optimal extreme-point solution is found. The method is generally implemented in two phases. In the first phase, an augmented system is initialized with an easily identifiable extreme-point solution using artificial variables to measure infeasibilities, and then optimized using the simplex algorithm with a view to obtaining an extreme-point solution to the augmented system that is feasible for the original system. If a solution without artificial variables cannot be found, the original linear program is infeasible. Otherwise, the second phase of the method uses the original problem formulation (without artificial variables) and the feasible extreme-point solution from the first phase and moves from that solution to a neighboring, or *adjacent*, solution. With each successive move to another extreme point, the objective function value improves (assuming non-degeneracy,

discussed in Section 3.2) until either: (i) the algorithm discovers a ray along which it can move infinitely far (to improve the objective) while still remaining feasible, in which case the problem is unbounded, or (ii) the algorithm discovers an extreme-point solution with an objective function value at least as good as that of any adjacent extreme-point solution, in which case that extreme point can be declared *an* optimal solution.

An optimal basis is both primal and dual feasible. In other words, the primal variable values calculated from the basis satisfy the constraints and nonnegativity requirements of (P_{LP}), while the dual variable values derived from the basis satisfy the constraints of (D_{LP}). The primal simplex method works by constructing a primal basic feasible solution, then working to remove the dual infeasibilities. The dual simplex method [13] works implicitly on the dual problem (D_{LP}) while operating on the constraints associated with the primal problem (P_{LP}). It does so by constructing a dual basic feasible solution, and then working to remove the primal infeasibilities. In that sense, the two algorithms are symmetric. By contrast, one can also explicitly solve the dual problem (D_{LP}) by operating on the dual constraint set with either the primal or dual simplex method. In all cases, the algorithm moves from one adjacent extreme point to another to improve the objective function value (assuming non-degeneracy) at each iteration.

Primal simplex algorithm:

We give the steps of the revised simplex algorithm, which assumes that we have obtained a basis, B, and a corresponding initial basic feasible solution, x_B, to our system as given in (P_{LP}). Note that the primal simplex method consists of an application of the algorithm to obtain such a feasible basis (phase I), and a subsequent application of the simplex algorithm with the feasible basis (phase II).

We define c_B and A_B as, respectively, the vector of objective coefficients and matrix coefficients associated with the basic variables, ordered as the variables appear in the basis. The nonbasic variables belong to the set N, and are given by $\{1, 2, \ldots, n\} - \{B\}$.

The revised simplex algorithm mitigates the computational expense and storage requirements associated with maintaining an entire simplex tableau, i.e., a matrix of A, b, and c components of a linear program, equivalent to the original but relative to a given basis, by computing only essential tableau elements. By examining the steps of the algorithm in the following list, the practitioner can often identify the aspects of the model that dominate the revised simplex method computations, and thus take suitable remedial action to reduce the run time.

1. **Backsolve** Obtain the dual variables by solving the linear system $y^T A_B = c_B^T$, where A_B is represented by an LU factorization. The LU factorization is a product of a lower triangular and upper triangular matrix that is computed through a sequence of pivoting operations analogous to the operations used to compute the inverse of a matrix. Most simplex algorithm implementations compute and maintain an LU factorization rather than a basis inverse because the former is sparser and can be computed in a more numerically stable manner. See [14] for details. Given a factorization $A_B = LU$, it follows that $A_B^{-1} = U^{-1}L^{-1}$, so the backsolve is equivalent to solving

$$y^T = c_B^T {A_B}^{-1} = c_B^T U^{-1} L^{-1}. \tag{1}$$

This can be calculated efficiently by solving the following sparse triangular linear systems (first, by computing the solution of a linear system involving U, and then using the result to solve a linear system involving L); this computation can be performed faster than computing y^T as a matrix product of c^T and ${A_B}^{-1}$.

$$p^T U = c_B^T$$
$$y^T L = p^T.$$

2. **Pricing** Calculate the reduced costs $\bar{c}_N^T = c_N^T - y^T A_N$, which indicate for each nonbasic variable the rate of change in the objective with a unit increase in the corresponding variable value from zero.
3. **Entering variable selection** Pick the entering variable x_t and associated incoming column A_t from the set of nonbasic variable indices N with $\bar{c}_N^T < 0$. If $\bar{c}_N^T \geq 0$, stop with an optimal solution $x = (x_B, 0)$.
4. **Forward solve** Calculate the corresponding incoming column w relative to the current basis matrix by solving $A_B w = A_t$.
5. **Ratio test** Determine the amount by which the value of the entering variable can increase from zero without compromising feasibility of the solution, i.e., without forcing the other basic variables to assume negative values. This, in turn, determines the position r of the outgoing variable in the basis and the associated index on the chosen variable $x_j, j \in \{1, 2, \ldots, n\}$. Call the outgoing variable in the rth position x_{j_r}. Then, such a variable is chosen as follows: $r = \text{argmin}_{i:w_i>0} \frac{x_{j_i}}{w_i}$.
   Let $\theta = \min_{i:w_i>0} \frac{x_{j_i}}{w_i}$.
   If there exists no i such that $w_i > 0$, then θ is infinite; this implies that regardless of the size of the objective function value given by a feasible solution, another feasible solution with a better objective function value always exists. The extreme-point solution given by $(x_B, 0)$, and the direction of unboundedness given by the sum of $(-w, 0)$ and the unit vector e_t combine to form the direction of unboundedness. Hence, stop because the linear program is unbounded. Otherwise, proceed to Step **6**.
6. **Basis update** Update the basis matrix A_B and the associated LU factorization, replacing the outgoing variable x_{j_r} with the incoming variable x_t. Periodically refactorize the basis matrix A_B using the factorization given above in Step **1** in order to limit the round-off error (see Section 3.1) that accumulates in the representation of the basis as well as to reduce the memory and run time required to process the accumulated updates.
7. **Recalculate basic variable values** Either update or refactorize. Most optimizers perform between 100 and 1000 updates between each refactorization.
   (a) **Update**: Let $x_t = \theta$; $x_i \leftarrow x_i - \theta \cdot w_i$ for $i \in \{B\} - \{t\}$.
   (b) **Refactorize:** Using the refactorization mentioned in Step **1** above, solve $A_B x_B = b$.
8. Return to Step **1**.

A variety of methods can be used to determine the incoming variable for a basis (see Step **3**) while executing the simplex algorithm. One can inexpensively select the incoming variable using *partial pricing* by considering a subset of nonbasic variables and selecting one of those with negative reduced cost. *Full pricing* considers the selection from all eligible variables. More elaborate variable selection schemes entail additional computation such as normalizing each negative reduced cost such that the selection of the incoming variable is based on a scale-invariant metric [15]. These more elaborate schemes can diminish the number of iterations needed to reach optimality but can also require more time per iteration, especially if a problem instance contains a large number of variables or if the A matrix is dense, i.e., it is computationally intensive to perform the refactorizations given in the simplex algorithm. Hence, if the decrease in the number of iterations required to solve the instance does not offset the increase in time required per iteration, it is preferable to use a simple pricing scheme. In general, it is worth considering non-default variable selection schemes for problems in which the number of iterations required to solve the instance exceeds three times the number of constraints.

Degeneracy in the simplex algorithm occurs when a basic variable assumes a value of zero as it enters the basis. In other words, the value θ in the minimum ratio test in Step **5** of the

simplex algorithm is zero. This results in iterations in which the objective retains the same value, rather than improving. Theoretically, the simplex algorithm can cycle, revisiting bases multiple times with no improvement in the objective. While cycling is primarily a theoretical, rather than a practical, issue, highly degenerate LPs can generate long, acyclic sequences of bases that correspond to the same objective, making the problem more difficult to solve using the simplex algorithm.

While space considerations preclude us from giving an analogous treatment of the dual simplex method [16], it is worth noting that the method is very similar to that of the primal simplex method, only preserving dual feasibility while iterating towards primal feasibility, rather than vice versa. The dual simplex algorithm begins with a set of nonnegative reduced costs; such a set can be obtained easily in the presence of an initial basic, dual-feasible solution or by a method analogous to Phase I of the primal simplex method. The primal variable values, x, are checked for feasibility, i.e., nonnegativity. If they are nonnegative, the algorithm terminates with an optimal solution; otherwise, a negative variable is chosen to exit the basis. Correspondingly, a minimum ratio test is performed on the quotient of the reduced costs and row associated with the exiting basic variable relative to the current basis (i.e., the simplex tableau row associated with the exiting basic variable). The ratio test either detects primal infeasibility or identifies the incoming basic variable. Finally, a basis update is performed on the factorization [17].

2.2. Interior point algorithms

The earliest interior point algorithms were the affine scaling algorithm proposed by Dikin [18] and the logarithmic barrier algorithm proposed by Fiacco and McCormick [19]. However, at that time, the potential of these algorithms for efficiently solving large-scale linear programs was largely ignored. The ellipsoid algorithm, proposed by Khachian [20], established the first polynomial time algorithm for linear programming. But, this great theoretical discovery did not translate to good performance on practical problems. It was not until Karmarkar's projective method [21] had shown great practical promise and was subsequently demonstrated to be equivalent to the logarithmic barrier algorithm [22], that interest in these earlier interior point algorithms increased. Subsequent implementations of various interior point algorithms revealed primal–dual logarithmic barrier algorithms as the preferred variant for solving practical problems [23].

None of these interior point algorithms or any of their variants uses a basis. Rather, the algorithm searches through the interior of the feasible region, avoiding the boundary of the constraint set until it finds an optimal solution. Each variant possesses a different means for determining a search direction. However, all variants fundamentally rely on centering the current iterate, computing an improving search direction, moving along it for a given *step size* short enough that the boundary is not reached (until optimality), and then re-centering the iterate.

While not the most efficient in practice, Dikin's Primal Affine Scaling Algorithm provides the simplest illustration of the computational steps of these interior point algorithms. Therefore, we describe Dikin's Algorithm in detail below. Lustig et al. [24] contains a more detailed description of the more frequently implemented primal–dual logarithmic barrier algorithm.

The kth iteration of Dikin's algorithm operates on the linear program (P_{LP}), along with a feasible interior point solution $x_k > 0$. The algorithm centers the feasible interior point by rescaling the variables based on the values of x_k, computes a search direction by projecting the steepest descent direction onto the null space of the rescaled constraint matrix, and moves in the search direction while ensuring that the new scaled solution remains a feasible interior point. The algorithm then unscales the solution, resulting in a new iterate, x_{k+1}. Following the unscaling, the algorithm performs a convergence test for optimality on x_{k+1}. If x_{k+1} is not optimal, the algorithm repeats its steps using x_{k+1} as the feasible interior point solution. The following steps provide the mathematical details.

Interior point algorithm with affine scaling:

1. **Centering** Let $D = Diag(x_k)$. Rescale the problem to center the current interior feasible solution by letting $\hat{A} = AD$, $\hat{c}^T = c^T D$. Hence, $\hat{x}_k = D^{-1}x_k = e$, the vector consisting of all 1's. Note that $\hat{A}\hat{x}_k = b$.
2. **Search Direction Computation** For the rescaled problem, project the steepest descent direction $-\hat{c}^T$ onto the null space of the constraint matrix $\hat{A}$, resulting in the search direction $p_k = -(I - \hat{A}^T(\hat{A}\hat{A}^T)^{-1}\hat{A})\hat{c}$.
3. **Step Length** Add a positive multiple θ of the search direction to p_k, the scaled interior feasible point, by computing $\hat{x}_{k+1} = e + \theta p_k$. If $p_k \geq 0$, then $\hat{x}_{k+1}$, and hence x_{k+1}, can increase without bound; stop the algorithm with an unbounded solution. Otherwise, because $\hat{A}p_k = 0$, $\hat{A}x_{k+1} = b$. Therefore, θ must be chosen to ensure that $\hat{x}_{k+1} > 0$. For any constant α such that $0 < \alpha < 1$, the update $\hat{x}_{k+1} = e - \left(\frac{\alpha}{\min_j p_k[j]}\right) p_k$ suffices.
4. **Optimality Test** Unscale the problem, setting $x_{k+1} = D\hat{x}_{k+1}$. Test x_{k+1} for optimality by checking whether $\|x_{k+1} - x_k\|$ is suitably small. If x_{k+1} is optimal, stop the algorithm. Otherwise, return to Step 1 with feasible interior point solution x_{k+1}.

The calculation of $p_k = -(I - \hat{A}^T(\hat{A}\hat{A}^T)^{-1}\hat{A})\hat{c}$ (Step 2) is the most time consuming operation of this algorithm. First, one must perform the matrix vector multiplication $v = \hat{A}\hat{c}$. Then, one must compute the solution w to the linear system of equations $(\hat{A}\hat{A}^T)w = v$. This step typically dominates the computation time of an iteration. Subsequently, one must perform a second matrix vector multiplication, $\hat{A}^T w$, then subtract $\hat{c}$ from the result.

The simplex algorithm creates an $m \times m$ basis matrix of left-hand-side coefficients, A_B, which is invertible. By contrast, interior point algorithms do not maintain a basis matrix. The matrix AA^T is not guaranteed to be invertible unless A has full rank. Fortunately, in the case of solving an LP, full rank comes naturally, either through removal of dependent constraints during presolve, or because of the presence of slack and artificial variables in the constraint matrix.

Other interior point algorithms maintain feasibility by different means. Examples include applying a logarithmic barrier function to the objective rather than explicitly projecting the search direction onto the null space of the rescaled problem, and using a projective transformation instead of the affine transformation of Dikin's algorithm to center the iterate. Some also use the primal and dual constraints simultaneously, and use more elaborate methods to calculate the search direction in order to reduce the total number of iterations. However, in all such practical variants to date, the dominant calculation remains the solution of a system of linear equations similar to the form $(\hat{A}\hat{A}^T)w = v$, as in Dikin's algorithm. As of the writing of this paper, the primal–dual barrier algorithm, combined with Mehrotra's predictor–corrector method [25], has emerged as the method of choice in most state-of-the-art optimizers.

Because the optimal solutions to linear programs reside on the boundary of the feasible region, interior point algorithms cannot move to the exact optimal solution. They instead rely on convergence criteria. Methods to identify an optimal basic solution from the convergent solution that interior point algorithms provide have been developed [26]. A procedure termed *crossover* can be invoked in most optimizers to transform a (typically near

$$\begin{pmatrix} x & 0 & 0 & 0 & 0 \\ x & x & 0 & 0 & 0 \\ x & 0 & x & 0 & 0 \\ x & 0 & 0 & x & 0 \\ x & 0 & 0 & 0 & x \end{pmatrix} * \begin{pmatrix} x & x & x & x & x \\ 0 & x & 0 & 0 & 0 \\ 0 & 0 & x & 0 & 0 \\ 0 & 0 & 0 & x & 0 \\ 0 & 0 & 0 & 0 & x \end{pmatrix} = \begin{pmatrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{pmatrix}$$

$$\begin{pmatrix} x & x & x & x & x \\ 0 & x & 0 & 0 & 0 \\ 0 & 0 & x & 0 & 0 \\ 0 & 0 & 0 & x & 0 \\ 0 & 0 & 0 & 0 & x \end{pmatrix} * \begin{pmatrix} x & 0 & 0 & 0 & 0 \\ x & x & 0 & 0 & 0 \\ x & 0 & x & 0 & 0 \\ x & 0 & 0 & x & 0 \\ x & 0 & 0 & 0 & x \end{pmatrix} = \begin{pmatrix} x & x & x & x & x \\ x & x & 0 & 0 & 0 \\ x & 0 & x & 0 & 0 \\ x & 0 & 0 & x & 0 \\ x & 0 & 0 & 0 & x \end{pmatrix}$$

Fig. 1. Let x denote an arbitrarily valued non-zero entry in the above matrices. The upper matrix product yields a completely dense result, while the lower matrix product yields a reasonably sparse result. The two products are merely reversals of each other.

optimal) interior solution to an optimal extreme-point solution. (In the case of a unique solution, the interior point method would converge towards the optimal extreme-point solution, i.e., a basic solution.) Crossover provides solutions that are easier to implement in practice. While crossover typically comprises a small percentage of the run time on most models, it can be time consuming, particularly when initiated on a suboptimal interior point solution with significant distance from an optimal solution. For primal–dual interior algorithms [27], the optimality criterion is usually based on a normalized duality gap, e.g., the quotient of the duality gap and primal objective (or dual objective since they are equal at optimality): $\frac{(c^T x - y^T b)}{c^T x}$.

In order to solve the linear system $(\hat{A}\hat{A}^T)w = v$ efficiently, most practical implementations maintain a Cholesky factorization $(\hat{A}\hat{A}^T) = \hat{L}\hat{L}^T$. The non-zero structure, i.e., the positions in the matrix in which non-zero elements of $\hat{A}\hat{A}^T$ lie, profoundly influences interior point algorithm run times. Since D is a diagonal matrix, $\hat{A}\hat{A}^T$ and AA^T have the same non-zero structure. For sparse matrices, the non-zero structure of AA^T depends heavily on the structure of the original matrix A. Note that even if the A matrix is sparse, the product of A and A^T can be dense. Consider the two matrix products in Fig. 1, in which x represents an arbitrarily valued non-zero matrix entry. The first product of A and A^T leads to a fully dense matrix (barring unlikely cancellation of terms in some entries), while the second product, which is simply a reversal of the two matrices A and A^T, remains relatively sparse.

A significant amount of work involves determining how to order A so that the Cholesky factor of AA^T is as sparse as possible. Although the non-zero values of $\hat{A}$ change from one iteration to the next, the non-zero structure of AA^T remains unchanged throughout the algorithm. So, interior point algorithm implementations can perform a large part of the computation that involves AA^T at the start of the algorithm by computing a symbolic Cholesky factorization of AA^T to provide the static non-zero structure of a lower triangular matrix L such that: $AA^T = LL^T$. This avoids recomputing $(\hat{A}\hat{A}^T)^{-1}$ from scratch during each iteration and removes the need to maintain any inverse at all. Rather, the implementation uses the Cholesky factorization to solve the system of equations $(\hat{A}\hat{A}^T)w = v$. At each iteration, the algorithm must update the individual values of the Cholesky factorization, but the non-zero structure of the factorization is known from the initial symbolic factorization.

2.3. Algorithm performance contrast

The linear programming algorithms we have discussed perform differently depending on the characteristics of the linear programs on which they are invoked. Although we introduce (P_{LP}) in standard form with equality constraints, yielding an $m \times n$ system with m equality constraints and n variables, we assume for the following discussion that our linear program is given as naturally formulated, i.e., with a mixture of equalities and inequalities, and, as such, contains m equality and inequality constraints and n variables.

The solution time for simplex algorithm iterations is more heavily influenced by the number of constraints than by the number of variables. This is because the row-based calculations in the simplex algorithm involving the $m \times m$ basis matrix usually comprise a much larger percentage of the iteration run time than the column-based operations. One can equivalently solve either the primal or the dual problem, so the practitioner should select the one that most likely solves fastest. Some optimizers have the ability to create the dual model before or after presolving a linear program, and to use internal logic for determining when to solve the explicit dual. The aspect ratio, $\frac{n}{m}$, generally indicates whether solving the primal or the dual problem, and/or solving the problem with the primal or the dual simplex method is likely to be more efficient. If $m \ll n$, it is more expedient to preserve the large aspect ratio. In this case, the primal simplex algorithm with partial pricing is likely to be effective because reduced costs on a potentially small subset of the nonbasic variables need to be computed at each iteration. By contrast, the dual simplex algorithm must compute a row of the simplex tableau during the ratio test that preserves dual feasibility. This calculation in the dual simplex algorithm involves essentially the same effort as full pricing in the primal simplex algorithm. This can consume a large portion of the computation time of each iteration. For models with $m \gg n$, an aspect ratio of 0.5 or smaller indicates that solving the dual explicitly yields faster performance than the primal, though solving the dual explicitly can also be faster even if the aspect ratio is greater than 0.5. Also, under primal degeneracy, implicitly solving the dual problem via the dual simplex method or solving the explicit dual can dramatically reduce the iteration count.

Regarding interior point algorithms, they may be effective when $m \ll n$. And, when m is relatively small, AA^T has only m rows and m columns, which gives the interior point algorithm a potential advantage over simplex algorithms, even when AA^T is relatively dense. For $m \gg n$, the interior point algorithm applied to the dual problem has potential to do very well. If A has either dense rows or dense columns, choosing between solving the primal and the dual affects only the interior point algorithm performance. Neither the primal nor the dual simplex algorithm and the associated factorized basis matrices has an obvious advantage over the other on an LP (or its dual) with dense rows or columns. However, in the interior point method, if AA^T is dense, and if the Cholesky factor is also dense, explicitly solving the dual model might produce faster run time performance. Dense columns in the primal problem, which frequently result in a dense AA^T matrix, become dense rows in the dual problem, which are less likely to result in a dense AA^T matrix (see Fig. 1). An A matrix which averages more than ten non-zero elements per column (regardless of the values of m and n), can make problems difficult to solve.

Most state-of-the-art optimizers can handle a modest number of dense columns by representing AA^T as the sum of outer products of the columns of A and separating the sum of the outer products of the dense columns. See [27] for more details. However, this approach can exhibit numerical instability (see Section 3.1) as the number of dense columns increases, so applying an interior point method to the explicit dual problem may be more effective.

Iteration Log #1 sheds light on the density of AA^T, as well as the additional non-zero matrix elements introduced during the factorization. The most important information in this output is the number of non-zeros in the lower triangle of AA^T and the total number of non-zeros in the Cholesky factor. An interior point algorithm is less likely to outperform the simplex algorithm if the number

Iteration Log #1

```
Reduced LP has 17645 rows, 35223 columns, and 93047 non-zeros.
Presolve time =    0.05 sec.
Parallel mode: using up to 4 threads for barrier.
***NOTE: Found 144 dense columns.
Number of non-zeros in lower triangle of A*A' = 31451
Using Approximate Minimum Degree ordering
Total time for automatic ordering = 0.00 sec.
Summary statistics for Cholesky factor:
  Threads                  = 4
  Rows in Factor           = 17789
  Integer space required   = 70668
  Total non-zeros in factor = 116782
  Total FP ops to factor   = 2587810
```

Iteration Log #2

```
Reduced LP has 17645 rows, 35223 columns, and 93047 non-zeros.
Presolve time =    0.05 sec.
Parallel mode: using up to 4 threads for barrier.
Number of non-zeros in lower triangle of A*A' = 153530245
Using Approximate Minimum Degree ordering
Total time for automatic ordering = 11.88 sec.
Summary statistics for Cholesky factor:
  Threads                  = 4
  Rows in Factor           = 17645
  Integer space required   = 45206
  Total non-zeros in factor = 153859571
  Total FP ops to factor   = 1795684140275
```

Iteration Log #3

```
Reduced LP has 79972 rows, 404517 columns, and 909056 non-zeros.
Presolve time =    1.57 sec.
Parallel mode: using up to 16 threads for barrier.
Number of non-zeros in lower triangle of A*A' = 566986
Using Nested Dissection ordering
Total time for automatic ordering = 9.19 sec.
Summary statistics for Cholesky factor:
  Threads                  = 16
  Rows in Factor           = 79972
  Integer space required   = 941792
  Total non-zeros in factor = 20737911
  Total FP ops to factor   = 24967192039
```

of non-zeros in the lower triangle of AA^T is much larger than the number of non-zeros in A, or if the number of non-zeros in the resulting Cholesky factor grows dramatically relative to the number of non-zeros in AA^T. Not surprisingly, then, the output in Iteration Log #1 suggests that an interior point algorithm is likely to outperform the simplex algorithm, since both the density of AA^T and the fill-in from the resulting factorization are quite modest. That is indeed true for this model (south31, a publicly available model from http://www.sztaki.hu/~meszaros/public_ftp/lptestset/misc/). CPLEX's barrier method solves the model in just under one second, in contrast to 7 and 16 seconds for the primal and dual simplex methods, respectively.

Now consider Iteration Log #2 which represents, in fact, output for the same model instance as considered in Iteration Log #1, but with dense column handling disabled. The A matrix has 93,047 non-zeros, while the lower triangle of AA^T from which the Cholesky factor is computed has over 153 million non-zeros, which represents a huge increase. The additional fill-in associated with the factorization is relatively negligible, with 153,859,571 − 153,530,245 = 329,326 additional non-zeros created.

The barrier algorithm must now perform calculations with a Cholesky factor containing over 153 million non-zeros instead of fewer than 120,000 non-zeros with the dense column handling. This increases the barrier run time on the model instance from less than one second to over 19 minutes.

Iteration Log #3 illustrates the case in which the non-zero count of AA^T (566,986) is quite modest relative to the non-zero count in A; however, the subsequent fill-in while computing the

Iteration Log #4

```
Reduced LP has 9652 rows, 9224120 columns, and 348547188 non-zeros.
Presolve time =  268.99 sec.
Parallel mode: using up to 16 threads for barrier.
Number of non-zeros in lower triangle of A*A' = 16747402
Using Nested Dissection ordering
Total time for automatic ordering = 409.44 sec.
Summary statistics for Cholesky factor:
  Threads                  = 16
  Rows in Factor           = 9652
  Integer space required   = 256438
  Total non-zeros in factor = 36344797
  Total FP ops to factor   = 171936037267
```

Cholesky factor is significant, as can be seen from the final non-zero count, which exceeds 20 million. And, indeed, the barrier algorithm requires more than 4 times as long as the dual simplex method to solve this model instance.

Finally, Iteration Log #4 illustrates the potential advantage of an interior point method over the simplex methods as the problem size grows very large. Due to its polynomial behavior, an interior point method's iteration count tends to increase slowly as problem size increases. Iteration Log #4 contains results for the zib05 model, available in MPS format at http://miplib.zib.de/contrib/miplib2003-contrib/IPWS2008/. Despite over 9 million columns in A with an average density of over 35 non-zeros per column, the modest number of rows and level of fill-in when computing the Cholesky factor results in a manageable model, provided sufficient memory (about 20 GB) is available. Due to the increased memory needed to solve this model, this run was done on a machine with 4 2.9 GHz quad core Xeon chips and 128 GB of memory.

By contrast, the simplex methods must select a basis of 9652 columns among the 9.2 million available, making this model much more challenging. CPLEX's barrier algorithm can solve this model in about 3 hours, while neither the primal nor the dual simplex method solves the model on the same machine in 20 hours.

The simplex method that can start with a feasible basis has a performance advantage over the one that lacks a feasible basis. Although, at the time of this writing, interior point methods cannot take advantage of a starting solution from a modified problem, the computational steps they perform parallelize better than those in the simplex algorithm. The simplex algorithm possesses a basis factorization whose non-zero structure changes at each iteration, while the interior point algorithm maintains a factorization with static non-zero structure throughout the algorithm that is more amenable to parallelization. The ability to parallelize an algorithm can produce faster run time performance.

Both the implicit dual, i.e., the dual simplex algorithm, and solving the dual explicitly, offer the advantages of (i) better performance in the presence of primal degeneracy, and (ii) a more easily available feasible starting basis on practical models. In many practical cases, dual feasibility may be easier to obtain than primal feasibility. If, as is often the case in practical models involving costs, all values of c are nonnegative, then $y = 0$ is immediately dual feasible. In fact, this holds even if the dual constraints are a mixture of equalities and inequalities (in either direction).

The basis with which the simplex algorithm operates allows for "warm starts", i.e., the ability to take advantage of a solution as a starting point for a slightly modified problem. Such a problem can be found as a subproblem in an iterative technique in which new variables and/or constraints have been inserted. For the case in which the practitioner is interested in multiple, similar solves, the choice of the primal or the dual simplex method is influenced by how the modifications that generate the sequence of instances preserve primal or dual feasibility of the basis and associated solution from the previous solve. For example, adding columns to (P_{LP}) maintains primal feasibility, but typically compromises dual feasibility. One can set the variables associated with the newly added columns to nonbasic at zero and append those values to the values of the existing primal feasible solution. Therefore, after adding columns, the primal simplex method has an advantage over the dual simplex method because the optimal basis from the previous LP remains primal feasible, but not typically dual feasible. Dual feasibility would also be preserved if the additional columns corresponded to redundant constraints in the dual, but such additions are uncommon in practice. Analogously, adding rows to a dual feasible solution (which is equivalent to adding columns to the explicit dual LP) preserves dual feasibility but typically compromises primal feasibility, so the dual simplex method has an advantage over the primal simplex method. Similarly, one can examine the different types of problem modifications to an LP and determine whether the modification preserves primal or dual feasibility. Table 1 summarizes the different types of modifications to the primal problem, (P_{LP}), and whether they ensure preservation of primal or dual feasibility. Note that the preservation of primal feasibility for a row of the table does not imply the absence of dual feasibility for the given problem modification (or vice versa). For some problem modifications, additional conclusions about preservation of feasibility can be made with a closer examination of the nature of the change. For example, changing an objective coefficient that relaxes a dual constraint preserves dual feasibility

Table 1
Different types of modifications to (P_{LP}) influence primal or dual feasibility of the modified LP.

Problem modification	Primal or dual feasibility preserved?
Add columns	Primal feasibility
Add rows	Dual feasibility
Change objective function coefficients	Primal feasibility
Change right-hand-side coefficients	Dual feasibility
Change matrix coefficients of basic variables	Neither
Change matrix coefficients of nonbasic variables	Primal feasibility

Table 2
Under various circumstances, different methods have a greater chance of faster solution time on a linear programming problem instance.

Characteristic	Recommended method
$m \ll n$	Primal simplex or interior point on primal problem
$m \gg n$	Primal simplex or interior point on dual problem
Dense rows in A matrix	Solve primal problem if using interior point
Dense columns in A matrix	Solve dual problem if using interior point
Availability of parallel hardware	Interior point
Multiple solves on similar instances necessary	Primal or dual simplex
Availability of a primal or dual feasible basis	Primal or dual simplex, respectively
Minimization with nonnegative c	Dual simplex as dual feasible basis is available

when the associated variable is nonbasic, but can result in a dual infeasible basis if said variable is basic.

Table 2 summarizes guidelines for the circumstances under which one should use primal simplex, dual simplex, or interior point method; the characteristics we present in this table are not mutually exclusive. For example, an LP may have $m \ll n$ but also have dense columns in the A matrix. Table 2 recommends either the primal simplex method on the primal problem, or an interior point method on the primal or dual problem. However, the dense columns potentially hinder the interior point method on the primal problem, while the large aspect ratio is potentially problematic for the dual problem. By contrast, there are no obvious limitations to primal simplex method performance. Therefore, while one should consider all of these recommendations, the primal simplex method on the primal LP has the most promise in this example.

3. Guidelines for successful algorithm performance

3.1. Numerical stability and Ill conditioning

Because most commonly used computers implement floating point computations in finite precision, arithmetic calculations such as those involved in solving linear programming problems can be prone to inaccuracies due to round-off error. Round-off error can arise from numerical instability or ill conditioning. In general terms, ill conditioning pertains to the situation in which a small change to the input can result in a much larger change to the output in models or systems of equations (linear or otherwise). Ill conditioning can occur under perfect arithmetic as well as under finite precision computing. Numerical stability (or lack thereof) is a characteristic of procedures and algorithms implemented under finite precision. A procedure is numerically stable if its backward error analysis results in small, bounded errors on all data instances, i.e., if a small, bounded perturbation to the model would make the computed solution to the unperturbed model an exact solution. Thus, numerical instability does not imply ill conditioning, nor does ill conditioning imply numerical instability. But, a numerically unstable algorithm introduces larger perturbations into its calculations than its numerically stable counterpart; this can lead to larger errors in the final computed solution if the model is ill conditioned. See [28] for more information on the different types of error analysis and their relationships to ill conditioning.

The practitioner cannot always control the floating point implementations of the computers on which he works and, hence, how arithmetic computations are done. As of this writing, exact floating point calculations can be done, but these are typically done in software packages such as Maple [29] and Mathematica [30], which are not large-scale linear programming solvers. The QSopt optimizer [31] reflects significant progress in exact linear programming, but even this solver still typically performs some calculations in finite precision. Regardless, the practitioner can and should be aware of input data and the implications of using an optimization algorithm and a floating point implementation on a model instance with such data. To this end, let us consider the derivation of the condition number of a square matrix, and how ill conditioning can affect the optimization of linear programs on finite-precision computers.

Consider a system of linear equations in standard form, $A_B x_B + A_N x_N = b$, where B constitutes the set of basic variables, N constitutes the set of non-basic variables, and A_B and A_N are the corresponding left-hand-side basic and non-basic matrix columns, respectively. Equivalently, x_B and x_N represent the vectors of basic and non-basic decision variables, respectively. We are interested in solving (P_{LP}), whose constraints we can rewrite as follows:

$$A_B x_B = b - A_N x_N = b. \tag{2}$$

Note that in (P_{LP}), all variables have lower bounds of zero and infinite upper bounds. Therefore, all nonbasic variables are zero and $A_N x_N = 0$. By contrast, if some of the variables have finite non-zero lower and/or upper bounds, then variables at these bounds can also be nonbasic and not equal to zero. Also note that Eq. (2) corresponds to Step 7b of the previously provided description of the primal simplex algorithm. In addition, Steps 1 and 4 solve similar systems of linear equations involving the basis matrix, A_B.

The exact solution of Eq. (2) is given by:

$$x_B = A_B^{-1} b. \tag{3}$$

Consider a small perturbation, Δb, to the right hand side of Eq. (2). We wish to assess the relation between Δb and the corresponding change Δx_B to the computed solution of the perturbed system of equations:

$$A_B(x_B + \Delta x_B) = b + \Delta b. \tag{4}$$

The exact solution of this system of Eq. (4) is given by:

$$(x_B + \Delta x_B) = A_B^{-1}(b + \Delta b). \tag{5}$$

Subtracting Eq. (3) from those given in (5), we obtain:

$$\Delta x_B = A_B^{-1} \Delta b. \tag{6}$$

Applying the Cauchy–Schwarz inequality to Eq. (6), we obtain:

$$\|\Delta x_B\| \le \|A_B^{-1}\| \|\Delta b\|. \tag{7}$$

In other words, the expression (7) gives an upper bound on the maximum absolute change in x_B relative to that of b. Similarly, we can get a relative change in x_B by applying the Cauchy–Schwarz inequality to Eq. (2):

$$\|b\| \le \|A_B\| \|x_B\|. \tag{8}$$

Multiplying the left and right hand sides of (7) and (8) together, and rearranging terms:

$$\frac{\|\Delta x_B\|}{\|x_B\|} \le \|A_B\| \|A_B^{-1}\| \left(\frac{\|\Delta b\|}{\|b\|} \right). \tag{9}$$

From (9), we see that the quantity $\kappa = \|A_B\|\|A_B^{-1}\|$ is a scaling factor for the relative change in the solution, $\frac{\|\Delta x_B\|}{\|x_B\|}$, given a relative change in the right hand side, $\frac{\|\Delta b\|}{\|b\|}$. The quantity $\kappa\left(\frac{\|\Delta b\|}{\|b\|}\right)$ provides an upper bound on the relative change in the computed solution, $\frac{\|\Delta x_B\|}{\|x_B\|}$, for a given relative change in the right hand side, $\left(\frac{\|\Delta b\|}{\|b\|}\right)$. Recall that ill conditioning in its most general sense occurs when a small change in the input of a system leads to a large change in the output. The quantity κ defines the *condition number* of the matrix A_B and enables us to assess the ill conditioning associated with the system of equations in (2). Larger values of κ imply greater potential for ill conditioning in the associated square linear system of equations by indicating a larger potential change in the solution given a change in the (right-hand-side) inputs. Because both the simplex and interior point algorithms need to solve square systems of equations, the value of κ can help predict how ill conditioning affects the computed solution of a linear program. Note that the condition number κ has the same interpretation when applied to small perturbations in A_B.

In practice, perturbations Δb can occur due to (i) finite precision in the computer on which the problem is solved, (ii) round-off error in the calculation of the input data to the problem, or (iii) round-off error in the implementation of the algorithm used to solve the problem. Note that these three issues are related. Input data with large differences in magnitude, even if computed precisely, require more shifting of exponents in the finite precision computing design of most computers than data with small differences in magnitude. This typically results in more round-off error in computations involving numbers of dramatically different orders of magnitude. The increased round-off error in these floating point calculations can then increase the round-off error that accumulates in the algorithm implementation. Most simplex algorithm implementations are designed to reduce the level of such round-off error, particularly as it occurs in the ratio test and LU factorization calculations. However, for some LPs, the round-off error remains a problem, regardless of the efforts to contain it in the implementation. In this case, additional steps must be taken to reduce the error by providing more accurate input data, by improving the conditioning of the model, or by tuning algorithm parameters. While this discussion is in the context of a basis matrix of the simplex algorithm, the calculations in Eqs. (3)–(9) apply to any square matrix and therefore also apply to the system of equations involving AA^T in the barrier algorithm.

Let us consider the condition number of the optimal basis to a linear program. Given the typical machine precision of 10^{-16} for double precision calculations and Eq. (9) that defines the condition number, a condition number value of 10^{10} provides an important threshold value. Most state-of-the-art optimizers use default feasibility and optimality tolerances of 10^{-6}. In other words, a solution is declared feasible when solution values that violate the lower bounds of 0 in (P_{LP}) do so by less than the feasibility tolerance. Similarly, a solution is declared optimal when any negative reduced costs in (P_{LP}) are less (in absolute terms) than the optimality tolerance. Because of the values of these tolerances, condition numbers of 10^{10} or greater imply a level of ill conditioning that could cause the implementation of the algorithm to make decisions based on round-off error. Because (9) is an inequality, a condition number of 10^{10} does not guarantee ill conditioning, but it provides guidance as to when ill conditioning is likely to occur.

Round-off error associated with finite precision implementations depends on the order of magnitude of the numbers involved in the calculations. Double precision calculations involving numbers with orders of magnitude larger than 10^0 can introduce a round-off error that is larger than the machine precision. However, because machine precision defines the smallest value that distinguishes two numbers, calculations involving numbers with smaller orders of magnitude than 10^0 still possess round-off errors at the machine precision level. For example, for floating point calculations involving at least one number on the order of 10^5, round-off error due to machine precision can be on the order of $10^5 * 10^{-16} = 10^{-11}$. Thus, round-off error for double precision calculations is relative, while most optimizers use absolute tolerances for assessing feasibility and optimality. State-of-the-art optimizers typically scale the linear programs they receive to try to keep the round-off error associated with double precision calculations close to the machine precision. Nonetheless, the practitioner can benefit from formulating LPs that are well scaled, avoiding mixtures of large and small coefficients that can introduce round-off errors significantly larger than machine precision. If this is not possible, the practitioner may need to consider solving the model with larger feasibility or optimality tolerances than the aforementioned defaults of 10^{-6}.

Eqs. (3)–(9) above are all done under perfect arithmetic. Finite-precision arithmetic frequently introduces perturbations to data. If a perturbation to the data is on the order of 10^{-16} and the condition number is on the order of 10^{12}, then round-off error as large as $10^{-16} * 10^{12} = 10^{-4}$ can creep into the calculations, and linear programming algorithms may have difficulty distinguishing numbers accurately within its 10^{-6} default optimality tolerance. The following linear program provides an example of ill conditioning and round-off error in the input data:

$$\max x_1 + x_2 \tag{10}$$

$$\text{subject to } \frac{1}{3}x_1 + \frac{2}{3}x_2 = 1 \tag{11}$$

$$x_1 + 2x_2 = 3 \tag{12}$$

$$x_1, x_2 \geq 0. \tag{13}$$

Because Eqs. (11) and (12) are linearly dependent, $\{x_1, \ x_2\}$ cannot form a basis. Rather, x_1 is part of the optimal basis, along with an artificial variable, and the optimal solution under perfect precision in this case is $x_1^* = 3$ and $x_2^* = 0$. By contrast, imprecisely rounded input with coefficients on x_1 and x_2 of 0.333333 and 0.666667, respectively, in the first constraint produces no linear dependency in the constraint sets (11) and (12), allowing both x_1 and x_2 to be part of a feasible basis. Furthermore, the optimal solution under perfect precision, $x_1^* = 3$ and $x_2^* = 0$, is infeasible in (11) with the rounded coefficients (because $3 \times 0.333333 \neq 1$). Instead, a solution of $x_1^* = x_2^* = 1$ satisfies both (11) and (12) using the default feasibility tolerance of 10^{-6} and is, in fact, optimal in this case. The associated optimal basis is $\{x_1, \ x_2\}$, with condition number $8.0 * 10^6$. Using three more digits of accuracy on the coefficients of x_1 and x_2 in constraint (11) yields the correct optimal solution. The optimal basis now consists of x_1 and one of the artificial variables, resulting in a condition number of 8.0. However, the better approach is to avoid rounding (or approximating) altogether by appropriately scaling the problem (i.e., multiplying through by the denominator) such that the first constraint is expressed as $x_1 + 2x_2 = 3$. When the first constraint is expressed in this way, constraints (11) and (12) are obviously redundant, which simplex algorithm implementations can easily handle. By contrast, the representation using rounding yields a near-singular basis matrix, A_B.

When a linear program is poorly scaled or ill conditioned, the simplex algorithm may lose feasibility when solving the problem instance. Even if the instance remains feasible, the algorithm might try to refactorize $A_B = LU$ to regain digits of accuracy it may have lost due to round-off error in the basis updates. Iteration Log #5, involving a model with an objective to be maximized, illustrates an instance that loses feasibility. The log prints the objective function value each time the algorithm refactorizes,

rather than updates, A_B. The algorithm tries to refactorize the basis matrix three times in three iterations, and increases the Markowitz threshold to improve the accuracy with which these computations are done. Larger values of the Markowitz threshold impose tighter restrictions on the criteria for a numerically stable pivot during the computation of the LU factorization [14]. However, despite the increased accuracy in the factorization starting at iteration 6391, enough round-off error accumulates in the subsequent iterations so that the next refactorization, at iteration 6456, results in a loss of feasibility.

Iteration Log #5

```
Iter: 6389 Objective  = 13137.039899
Iter: 6390 Objective  = 13137.039899
Iter: 6391 Objective  = 13726.011591
Markowitz threshold set to 0.3.
Iter: 6456 Scaled infeas = 300615.030682
...
Iter: 6752 Scaled infeas = 0.000002
Iter: 6754 Objective = -23870.812630
```

Although the algorithm regains feasibility at iteration 6754, it spends an extra 298 (6754–6456) iterations doing so, and the objective function is much worse than the one at iteration 6389. Additional iterations are then required just to regain the previously attained objective value. So, even when numerical instability or ill conditioning does not prevent the optimizer from solving the model to optimality, it may slow down performance significantly. Improvements to the model formulation by reducing the source of perturbations and, if needed, changes to parameter settings can reduce round-off error in the optimizer calculations, resulting in smaller basis condition numbers and faster computation of optimal solutions.

As another caveat, the practitioner should avoid including data with meaningful values smaller than the optimizer's tolerances. Similarly, the practitioner should ensure that the optimizer's tolerances exceed the largest round-off error in any of the data calculations. The computer can only handle a fixed number of digits. Very small numerical values force the algorithm to make decisions about whether those smaller values are real or due to round-off error, and the optimizer's decisions can depend on the coordinate system (i.e., basis) with which it views the model. Consider the following feasibility problem:

$$c_1 : -x_1 + 24x_2 \leq 21 \tag{14}$$

$$-\infty < x_1 \leq 3 \tag{15}$$

$$x_2 \geq 1.00000008. \tag{16}$$

The primal simplex method concludes infeasibility during the presolve:

Iteration Log #6

```
CPLEX> primopt
Infeasibility row 'c1':  0 <= -1.92e-06.
Presolve time =    0.00 sec.
Presolve - Infeasible.
Solution time =    0.00 sec.
```

Turning presolve off causes the primal simplex method to arrive at a similar conclusion during the first iteration.

Iteration Log #7

```
Primal simplex - Infeasible:
Infeasibility =  1.9199999990e-06
Solution time =   0.00 sec.  Iterations = 0 (0)
CPLEX> display solution reduced -
Variable Name             Reduced Cost
x1                           -1.000000
x2                           24.000000
CPLEX> display solution slacks -
Constraint Name            Slack Value
slack c1                     -0.000002**
CPLEX> display solution basis variables -
There are no basic variables in the given range.
CPLEX> display solution basis slack -
Constraint 'c1' is basic.
```

The asterisks on the slack value for constraint c_1 signify that the solution violates the slack's lower bound of 0.

These two runs both constitute correct outcomes. In Iteration Log #6, CPLEX's presolve uses the variable bounds and constraint coefficients to calculate that the minimum possible value for the left hand side of constraint c_1 is $-3 + 24 * 1.00000008 = 21 + 1.92 * 10^{-6}$. This means that the left hand side must exceed the right hand side, and by a value of more than that of CPLEX's default feasibility tolerance of 10^{-6}. Iteration Log #7 shows that with presolve off, CPLEX begins the primal simplex method with the slack on constraint c_1 in the basis, and the variables x_1 and x_2 at their respective bounds of 3 and 1.00000008. Given this basis, the reduced costs, i.e., the optimality criterion from Phase I, indicate that there is no way to remove the infeasibility, so the primal simplex method declares the model infeasible. Note that most optimizers treat variable bound constraints separately from general linear constraints, and that a negative reduced cost on a variable at its upper bound such as x_1 indicates that decreasing that variable from its upper bound cannot decrease the objective. Now, suppose we run the primal simplex method with a starting basis of x_2, the slack variable nonbasic at its lower bound, and x_1 nonbasic at its upper bound. The resulting basic solution of $x_1 = 3$, $x_2 = 1$, slack on $c_1 = 0$ satisfies constraint c_1 exactly. The variable x_2 does not satisfy its lower bound of 1.00000008 exactly, but the violation is less than many optimizers' default feasibility tolerance of 10^{-6}. So, with this starting basis, an optimizer could declare the model feasible (and hence optimal, because the model has no objective function):

Iteration Log #8

```
Primal simplex - Optimal:
Objective = 0.0000000000e+00
Solution time = 0.00 sec.  Iterations = 0 (0)
```

In this example, were we to set the feasibility tolerance to 10^{-9}, we would have obtained consistent results with respect to both bases because the data do not possess values smaller than the relevant algorithm tolerance. Although the value of 0.00000008 is input data, this small numerical value could have just as easily been created during the course of the execution of the algorithm. This example illustrates the importance of verifying that the optimizer tolerances properly distinguish legitimate values from those arising from round-off error. When a model is on the edge of feasibility, different bases may prove feasibility or infeasibility relative to the optimizer's tolerances. Rather than relying on the optimizer to make such important decisions, the practitioner should ensure that the optimizer's tolerances are suitably set to

reflect the valid precision of the data values in the model. In the example we just examined, one should first determine whether the lower bound on x_2 is really 1.00000008, or if, in fact, the fractional part is round-off error in the data calculation and the correct lower bound is 1.0. If the former holds, the practitioner should set the optimizer's feasibility and optimality tolerances to values smaller than 0.00000008. If the latter holds, the practitioner should change the lower bound to its correct value of 1.0 in the model. In this particular example, the practitioner may be inclined to deduce that the correct value for the lower bound on x_2 is 1.0, because all other data in the instance are integers. More generally, examination of the possible round-off error associated with the procedures used to calculate the input data may help to distinguish round-off error from meaningful values.

One particularly problematic source of round-off error in the data involves the conversion of single precision values to their double precision counterparts used by most optimizers. Precision for an IEEE single precision value is $6*10^{-8}$, which is almost as large as many of the important default optimizer tolerances. For example, CPLEX uses default feasibility and optimality tolerances of 10^{-6}. So, simply *representing* a data value in single precision can introduce round-off error of at least $6 * 10^{-8}$, and additional single precision data calculations can increase the round-off error above the aforementioned optimizer tolerances. Hence, the optimizer may subsequently make decisions based on round-off error. Computing the data in double precision from the start will avoid this problem. If that is not possible, setting the optimizer tolerances to values that exceed the largest round-off error associated with the conversion from single to double precision provides an alternative.

All linear programming algorithms can suffer from numerical instability. In particular, the choice of primal or dual simplex algorithm does not affect the numerical stability of a problem instance because the LU factorizations are the same with either algorithm. However, the interior point algorithm is more susceptible to numerical stability problems because it tries to maintain an interior solution, yet as the algorithm nears convergence, it requires a solution on lower dimensional faces of the polyhedron, i.e., the boundary of the feasible region.

3.2. Degeneracy

Degeneracy in the simplex algorithm occurs when the value θ in the minimum ratio test in Step 5 of the simplex algorithm (see Section 2.1) is zero. This results in iterations in which the objective retains the same value, rather than improving. Highly degenerate LPs tend to be more difficult to solve using the simplex algorithm. Iteration Log #9 illustrates degeneracy: the nonoptimal objective does not change between iterations 5083 and 5968; therefore, the algorithm temporarily perturbs the right hand side or variable bounds to move away from the degenerate solution.

Iteration Log #9

```
Iter: 4751 Infeasibility = 8.000000
Iter: 4870 Infeasibility = 8.000000
Iter: 4976 Infeasibility = 6.999999
Iter: 5083 Infeasibility = 6.000000
Iter: 5191 Infeasibility = 6.000000

...

Iter: 5862 Infeasibility = 6.000000
Iter: 5968 Infeasibility = 6.000000
Perturbation started.
```

After the degeneracy has been mitigated, the algorithm removes the perturbation to restore the original problem instance. If the current solution is not feasible, the algorithm performs additional iterations to regain feasibility before continuing the optimization run. Although a pricing scheme such as Bland's rule can be used to mitigate cycling through bases under degeneracy, this rule holds more theoretical, than practical, importance and, as such, is rarely implemented in state-of-the-art optimizers. While such rules prevent cycles of degenerate pivots, they do not necessarily prevent long sequences of degenerate pivots that do not form a cycle but do inhibit primal or dual simplex method performance.

When an iteration log indicates degeneracy, first consider trying all other LP algorithms. Degeneracy in the primal LP does not necessarily imply degeneracy in the dual LP. Therefore, the dual simplex algorithm might effectively solve a highly primal degenerate problem, and vice versa. Interior point algorithms are not prone to degeneracy because they do not pivot from one extreme point to the next. Interior point solutions are, by definition, nondegenerate. If alternate algorithms do not help performance (perhaps due to other problem characteristics that make them disadvantageous), a small, random perturbation of the problem data may help. Primal degenerate problems can benefit from perturbations of the right hand side values, while perturbations of the objective coefficients can help on dual degenerate problems. While such perturbations do not guarantee that the simplex algorithm does not cycle, they frequently yield improvements in practical performance. Some optimizers allow the practitioner to request perturbations by setting a parameter; otherwise, one can perturb the problem data explicitly.

3.3. Excessive simplex algorithm iteration counts

As described in the previous section, degeneracy can increase simplex algorithm iteration counts. However, the simplex algorithm may exhibit excessive iterations (typically, at least three times the number of constraints) for other reasons as well. For some models, the algorithm may make inferior choices when selecting the entering basic variable. In such cases, more computationally elaborate selection schemes than partial or full pricing that compute additional information can reduce the number of iterations enough to outweigh any associated increase in time per iteration. Today's state-of-the-art optimizers typically offer parameter settings that determine the entering variable selection to the practitioner, and selections other than the default can significantly improve performance.

Steepest edge and Devex pricing are the most popular of these more informative selection rules. Steepest edge pricing computes the L2 norm of each nonbasic matrix column relative to the current basis. Calculating this norm explicitly at each iteration by performing a forward solve, as in Step 4 of the primal simplex algorithm, would be prohibitively expensive (with the possible exception of when a large number of parallel threads is available). However, such computation is unnecessary, as all of the Steepest edge norms can be updated at each iteration with two additional backward solves, using the resulting vectors in inner products with the nonbasic columns in a manner analogous to full pricing [32,33]. Devex pricing [34] estimates part of the Steepest edge update, removing one of the backward solves and one of the aforementioned inner products involving the nonbasic matrix columns. These methods can be implemented efficiently in both the primal and dual simplex algorithms.

The initial calculation of the exact Steepest edge norms can also be time consuming, involving a forward solve for each L2 norm or a backward solve for each constraint. Steepest edge variants try to reduce or remove this calculation by using estimates of the initial

Iteration Log #10

```
Problem '/ilog/models/lp/all/pilot87.sav.gz' read.

...

Iteration log . . .
Iteration:     1   Scaled dual infeas =             0.676305
Iteration:   108   Scaled dual infeas =             0.189480
Iteration:   236   Scaled dual infeas =             0.170966
...

Iteration:  8958   Dual objective     =           302.913722
Iteration:  9056   Dual objective     =           303.021157
Iteration:  9137   Dual objective     =           303.073444
Removing shift (4150).
Iteration:  9161   Scaled dual infeas = 0.152475
Iteration:  9350   Scaled dual infeas =             0.001941
Iteration:  9446   Scaled dual infeas =             0.000480
Iteration:  9537   Dual objective     =           299.891447
Iteration:  9630   Dual objective     =           301.051704
Iteration:  9721   Dual objective     =           301.277884
Iteration:  9818   Dual objective     =           301.658507
Iteration:  9916   Dual objective     =           301.702665
Removing shift (41).
Iteration: 10008   Scaled dual infeas = 0.000136
Iteration: 10039   Dual objective     =           301.678880
Iteration: 10140   Dual objective     =           301.710360
Removing shift (8).
Iteration: 10151   Objective     =           301.710354

Dual simplex - Optimal:  Objective =  3.0171035068e+02
Solution time =    6.38 sec.  Iterations = 10154 (1658)
```

norms that are easier to compute. After computing these initial estimates, subsequent updates to the norms are done exactly. By contrast, Devex computes initial estimates to the norms, followed by estimates of the norm updates as well.

Since the additional computations for Steepest edge comprise almost as much work as a primal simplex algorithm iteration, this approach may need a reduction in the number of primal simplex iterations of almost 50% to be advantageous for the algorithm. However, the Steepest edge norm updates for the dual simplex algorithm involve less additional computational expense [35]; in this case, a 20% reduction in the number of iterations may suffice to improve performance. Devex pricing can also be effective if it reduces iteration counts by 20% or more. State-of-the-art optimizers may already use some form of Steepest edge pricing by default. If so, Steepest edge variants or Devex, both of which estimate initial norms rather than calculate them exactly, may yield similar iteration counts with less computation time per iteration.

Model characteristics such as constraint matrix density and scaling typically influence the tradeoff between the additional computation time and the potential reduction in the number of iterations associated with Steepest edge, Devex or other entering variable selection schemes. The practitioner should keep this in mind when assessing the effectiveness of these schemes. For example, since these selection rules involve additional backward solves and pricing operations, their efficacy depends on the density of the constraint matrix A. Denser columns in A increase the additional computation time per iteration. Some optimizers have default internal logic to perform such assessments automatically and to use the selection scheme deemed most promising. Nonetheless, for LPs with excessive iteration counts, trying these alternate variable selection rules can improve performance relative to the optimizer's default settings.

Iteration Logs #6, #7 and #8 illustrate the importance of selecting optimizer tolerances to properly distinguish legitimate values from those arising from round-off error. In those iteration logs, this distinction is essential to determine if a model was infeasible or feasible. This distinction can also influence the number of simplex algorithm iterations, and proper tolerance settings can improve performance. In particular, many implementations of the simplex method use the Harris ratio test [34] or some variant thereof. Harris' method allows more flexibility in the selection of the outgoing variable in the ratio test for the primal or dual simplex method, but it does so by allowing the entering variable to force the outgoing variable to a value slightly below 0. The optimizer then shifts the variable lower bound of 0 to this new value to preserve feasibility. These bound shifts are typically limited to a tolerance value no larger than the optimizer's feasibility tolerance. While this offers advantages regarding more numerically stable pivots and less degeneracy, such violations must eventually be addressed since they can potentially create small infeasibilities. In some cases, performance can be improved by reducing the maximum allowable violation in the Harris ratio test.

Iteration Logs #10 and #11 illustrate how reducing the maximum allowable violation in the Harris ratio test can improve performance. The model, pilot87, is publicly available from

Iteration Log #11

```
New value for feasibility tolerance: 1e-09
New value for reduced cost optimality tolerance: 1e-09

...

Iteration log . . .
Iteration:     1   Scaled dual infeas =             0.676355
Iteration:   123   Scaled dual infeas =             0.098169
...

Removing shift (190).
Iteration:  9332   Scaled dual infeas =             0.000004
Iteration:  9338   Dual objective     =           301.710248

Dual simplex - Optimal:  Objective =  3.0171034733e+02
Solution time =    6.15 sec.  Iterations = 9364 (1353)
```

the NETLIB set of linear programs at http://www.netlib.org/lp/. Iteration Log #10 illustrates a run with default feasibility and optimality tolerances of 10^{-6}. However, because pilot87 contains matrix coefficients as small as 10^{-6}, the resulting bound violations and shifts allowed in the Harris ratio test can create meaningful infeasibilities. Hence, when CPLEX removes the shifted bounds starting at iteration 9161, it must repair some modest dual infeasibilities with additional dual simplex iterations. Subsequent iterations are performed with a reduced limit on the bound shift, but the removal of the additional bound shifts at iteration 10,008 results in some slight dual infeasibilities requiring additional iterations. Overall, CPLEX spends 993 additional dual simplex iterations after the initial removal of bound shifts at iteration 9161. Since this model has (apparently legitimate) matrix coefficients of 10^{-6}, the default feasibility and optimality tolerances are too large to enable the optimizer to properly distinguish legitimate values from round-off error. The bound shifts thus are large enough to create dual infeasibilities that require additional dual simplex iterations to repair.

By contrast, Iteration Log #11 illustrates the corresponding run with feasibility and optimality tolerances reduced to 10^{-9}. This enables CPLEX to distinguish the matrix coefficient of 10^{-6} as legitimate in the model. Thus, it uses smaller bound violations and shifts in the Harris ratio test. Therefore, it needs to remove the bound shifts once, and only requires 32 additional dual simplex iterations to prove optimality. While the overall reduction in run time with this change is a modest four percent, larger improvements are possible on larger or more numerically challenging models.

3.4. Excessive barrier algorithm iteration counts

Section 2.3 included a discussion of how the non-zero structure of the constraint matrix influences barrier and other interior point algorithms' time per iteration. On most LPs, the weakly polynomial complexity of the barrier algorithm results in a very modest number of iterations, even as the size of the problem increases. Models with millions of constraints and variables frequently solve in fewer than 100 iterations. However, because the barrier algorithm relies on convergence criteria, the algorithm may struggle to converge, performing numerous iterations with little or no improvement in the objective. Most barrier algorithm implementations include an adjustable convergence tolerance that can be used to determine when the algorithm should stop, and proceed to the crossover procedure to find a basic solution. For some models, increasing the barrier convergence tolerance avoids barrier iterations of little, if any, benefit to the crossover procedure. In such cases, a larger barrier convergence tolerance may save significant time when the barrier method run time dominates the crossover run time. By contrast, if the crossover time with default settings comprises a significant part of the optimization time and the barrier iteration count is modest, reducing the barrier convergence tolerance may provide the crossover procedure a better interior point with which to start, thus improving performance.

Iteration logs #12 and #13 provide an example in which increasing the barrier convergence tolerance improves performance. The model solved in the logs is Linf_520c.mps, publicly available at Hans Mittelmann's Benchmarks for Optimization Software website (http://plato.asu.edu/ftp/lptestset/). Because this model has some small coefficients on the order of 10^{-6}, the runs in these logs follow the recommendations in Section 3.1 and use feasibility and optimality tolerances of 10^{-8} to distinguish them from any meaningful values in the model.

Columns 2 and 3 of Iteration Log #12 provide the information needed to assess the relative duality gap $\frac{(c^T x - y^T b)}{c^T x}$ typically compared with the barrier convergence tolerance. Not surprisingly, the gap is quite large initially. However, the gap is much smaller by iteration 20. Modest additional progress occurs by iteration 85, but little progress occurs after that, as can be seen at iterations 125 and 175. In fact, at iteration 175, CPLEX's barrier algorithm has stopped improving relative to the convergence criteria, as can be seen by the slight increases in relative duality gap, primal bound infeasibility and dual infeasibility (in the second, third, fifth and sixth columns of the iteration log). CPLEX therefore initiates crossover using the solution from iteration 19. Thus, iterations 20 through 175 are essentially wasted. The solution values at iteration 19 still have a significant relative duality gap, so the crossover procedure finds a basis that requires additional simplex method iterations. Iteration Log #13 illustrates how, by increasing the barrier convergence tolerance from the CPLEX default of 10^{-8} to 10^{-4}, much of the long tail of essentially wasted iterations is removed. In this case, the optimizer does not need to restore solution values from an earlier iteration due to increases in the relative duality gap. Hence, the optimizer initiates crossover at a solution closer to optimality. The basis determined by crossover is much closer to optimal than the one in Iteration Log #12, resulting in very few additional simplex iterations to find an optimal basis. This reduction in simplex

Iteration Log #12

```
 Itn      Primal Obj        Dual Obj  Prim Inf Upper Inf  Dual Inf
   0   5.9535299e+02  -2.7214222e+13  4.05e+11  1.03e+12  3.22e+07
   1   1.8327921e+07  -2.4959566e+13  2.07e+10  5.30e+10  6.74e+04
   2   7.5190135e+07  -5.6129961e+12  4.68e+09  1.20e+10  1.09e+03
   3   2.8110662e+08  -2.2938869e+12  5.95e+07  1.52e+08  4.46e+02
   4   2.8257248e+08  -4.7950310e+10  1.28e+06  3.26e+06  3.53e+00
...
  19   2.0017972e-01   1.9840302e-01  1.37e-05  1.14e-06  1.51e-06
  20   2.0003146e-01   1.9848712e-01  1.73e-05  1.30e-06  1.65e-06
  21   1.9987509e-01   1.9854536e-01  1.62e-05  1.37e-06  1.71e-06
...
  80   1.9896330e-01   1.9865235e-01  1.35e-04  2.87e-06  2.33e-06
  81   1.9895653e-01   1.9865287e-01  1.50e-04  2.79e-06  2.32e-06
  82   1.9895461e-01   1.9865256e-01  1.52e-04  2.87e-06  2.44e-06
  83   1.9895075e-01   1.9865309e-01  1.69e-04  2.86e-06  2.38e-06
  84   1.9894710e-01   1.9865303e-01  1.70e-04  2.85e-06  2.39e-06
  85   1.9894158e-01   1.9865347e-01  1.68e-04  2.77e-06  2.40e-06
...
 123   1.9888124e-01   1.9865508e-01  4.81e-04  3.01e-06  2.31e-06
 124   1.9888029e-01   1.9865515e-01  4.46e-04  2.89e-06  2.37e-06
 125   1.9887997e-01   1.9865506e-01  4.35e-04  2.93e-06  2.43e-06
...
 170   1.9886958e-01   1.9865768e-01  9.10e-04  2.63e-06  2.37e-06
 171   1.9886949e-01   1.9865771e-01  8.96e-04  2.56e-06  2.33e-06
 172   1.9886939e-01   1.9865739e-01  8.15e-04  2.45e-06  2.28e-06
 173   1.9886934e-01   1.9865738e-01  8.19e-04  2.47e-06  2.26e-06
 174   1.9886931e-01   1.9865728e-01  7.94e-04  2.52e-06  2.29e-06
 175   1.9886927e-01   1.9865716e-01  7.57e-04  2.55e-06  2.31e-06
   *   2.0017972e-01   1.9840302e-01  1.37e-05  1.14e-06  1.51e-06
Barrier time =  869.96 sec.
Primal crossover.
  Primal:  Fixing 34010 variables.
    34009 PMoves:  Infeasibility  3.11238869e-08  Objective  2.00179717e-01
    32528 PMoves:  Infeasibility  4.93933590e-08  Objective  2.00033745e-01
    31357 PMoves:  Infeasibility  6.39691687e-08  Objective  2.00033745e-01
...
  Elapsed crossover time =   20.14 sec. (3600 PMoves)
     3021 PMoves:  Infeasibility  1.14510022e-07  Objective  2.00033487e-01
     2508 PMoves:  Infeasibility  9.65797950e-08  Objective  2.00033487e-01
 ...
  Primal:  Pushed 15282, exchanged 18728.
  Dual:  Fixing 15166 variables.
    15165 DMoves:  Infeasibility  1.03339605e+00  Objective  1.98870673e-01
   Elapsed crossover time =   27.38 sec. (14800 DMoves)
...
   Elapsed crossover time =   68.41 sec. (1400 DMoves)
        0 DMoves:  Infeasibility  1.00261077e+00  Objective  1.98865022e-01
  Dual:  Pushed 10924, exchanged 4.
...
Iteration log . . .
Iteration:     1    Scaled infeas =         28211.724670
Iteration:    31    Scaled infeas =         23869.881089
Iteration:   312    Scaled infeas =          4410.384413
...
Iteration:  6670    Objective     =             0.202403
Iteration:  6931    Objective     =             0.199894
Elapsed time = 1061.69 sec. (7000 iterations).
Removing shift (5).
Iteration:  7072    Scaled infeas =             0.000000
Total crossover time =  192.70 sec.
Total time on 4 threads = 1063.55 sec.

Primal simplex - Optimal:  Objective =  1.9886847000e-01
Solution time = 1063.55 sec.  Iterations = 7072 (5334)
```

Iteration Log #13

```
 Itn      Primal Obj        Dual Obj  Prim Inf Upper Inf  Dual Inf
   0   5.9535299e+02  -2.7214222e+13  4.05e+11  1.03e+12  3.22e+07
   1   1.8327921e+07  -2.4959566e+13  2.07e+10  5.30e+10  6.74e+04
   2   7.5190135e+07  -5.6129961e+12  4.68e+09  1.20e+10  1.09e+03
   3   2.8110662e+08  -2.2938869e+12  5.95e+07  1.52e+08  4.46e+02
   4   2.8257248e+08  -4.7950310e+10  1.28e+06  3.26e+06  3.53e+00
   5   1.3900254e+08  -6.1411363e+07  1.21e-03  9.44e-07  6.00e-02

...

  82   1.9895461e-01   1.9865256e-01  1.52e-04  2.87e-06  2.44e-06
  83   1.9895075e-01   1.9865309e-01  1.69e-04  2.86e-06  2.38e-06
  84   1.9894710e-01   1.9865303e-01  1.70e-04  2.85e-06  2.39e-06
Barrier time =  414.31 sec.

Primal crossover.
  Primal:  Fixing 33950 variables.
    33949 PMoves:  Infeasibility  4.36828532e-07  Objective  1.98868436e-01
    32825 PMoves:  Infeasibility  4.32016730e-07  Objective  1.98868436e-01

...

      128 PMoves:  Infeasibility  5.49582923e-07  Objective  1.98868436e-01
        0 PMoves:  Infeasibility  5.49534369e-07  Objective  1.98868436e-01
  Primal:  Pushed 13212, exchanged 20737.
  Dual:  Fixing 71 variables.
       70 DMoves:  Infeasibility  2.72300071e-03  Objective  1.98842990e-01
        0 DMoves:  Infeasibility  2.72073853e-03  Objective  1.98842990e-01
  Dual:  Pushed 63, exchanged 0.
Using devex.

Iteration log . . .
Iteration:     1    Objective     =            0.198868
Removing shift (45).
Iteration:     2    Scaled infeas =            0.000000
Iteration:     6    Objective     =            0.198868
Total crossover time =   38.39 sec.

Total time on 4 threads =  452.71 sec.

Primal simplex - Optimal:  Objective =  1.9886847000e-01
Solution time =  452.71 sec.  Iterations = 6 (4)
```

iterations, in addition to the reduced number of barrier iterations, improves the overall run time from 1063 seconds to 452 seconds.

By contrast, if the barrier iteration counts are small (e.g., 50 or fewer), show no long tail of iterations with little progress as seen in Log #12, yet exhibit significant crossover and simplex iterations, decreasing the barrier convergence tolerance, rather than increasing it, often improves performance.

3.5. Excessive time per iteration

Algorithms may require an unreasonable amount of time per iteration. For example, consider Iteration Log #14 in which 37,000 iterations are executed within the first 139 seconds of the run; at the bottom of the log, 1000 (140,000–139,000) iterations require about 408 (25,145.43–24,736.98) seconds of computation time.

Iteration Log #14

```
Elapsed time = 138.23 sec. (37000 iterations)
Iter: 37969 Infeasibility = 387849.999786
Iter: 39121 Infeasibility = 379979.999768
Iter: 40295 Infeasibility = 375639.999998
Elapsed time = 150.41 sec. (41000 iterations)
...

Elapsed time = 24318.58 sec. (138000 iterations)
Iter: 138958 Infeasibility = 23.754244
Elapsed time = 24736.98 sec. (139000 iterations)
Elapsed time = 25145.43 sec. (140000 iterations)
```

This slowdown in iteration execution can be due to denser bases and the associated denser factorization and solve times in the simplex algorithm, specifically in Steps 1, 4 and 7. In this case, the practitioner should try each LP algorithm and consider an alternate pricing scheme if using the simplex algorithm. The more computationally expensive gradient pricing schemes available in today's state-of-the-art optimizers calculate (exactly

or approximately) the norm of each nonbasic matrix column relative to the current basis. The most expensive calculations involve computing these norms exactly, while less expensive calculations involve progressively cheaper (but progressively less accurate) ways of estimating these exact norms [15]. All of these norm calculations involve extra memory, extra computation, and extra pricing operations.

Another reason for excessive time per iteration is the computer's use of virtual memory. A general rule requires one gigabyte of memory per million constraints of a linear program and even more if the A matrix is very dense or if there is a significantly larger number of columns (variables) than rows (constraints). Short of purchasing more memory, some optimizers support non-default settings that compress data and/or store data efficiently to disk. For example, CPLEX has a memory emphasis parameter whose overhead is mitigated by reducing the optimizer's reliance on the operating system's virtual memory manager. This can help preserve memory and ultimately lead to finding a good solution reasonably quickly.

While the change in density of the basis matrices used by the simplex method can alter the time per iteration, each iteration of the barrier algorithm and related interior point methods solves a linear system involving AA^T with constant non-zero structure. Hence, barrier algorithm memory usage and time per iteration typically exhibit little change throughout the run. Section 2.3 describes how the density of AA^T and the associated Cholesky factorization influence the time per barrier algorithm iteration. However, that section assumes that the density of the Cholesky factor was given, and identified guidelines for assessing when the density was favorable for the barrier algorithm. While state-of-the-art optimizers do offer some parameter settings that can potentially reduce the density of the Cholesky factor, the default settings are very effective regarding the sparsity of the Cholesky factor for a given non-zero structure of AA^T. Instead of trying to find a sparser Cholesky factor for a given AA^T, the practitioner can adjust the model formulation so that AA^T is sparser, resulting in a sparser Cholesky factor and faster barrier algorithm iterations. One example of reformulation involves splitting dense columns into sparser ones [36,37]. While this increases the number of constraints and variables in the model, those dimensions are not the source of the performance bottleneck. A larger model with a Cholesky factor that has more constraints but fewer non-zeros can result in faster performance. For example, if A_j is a dense matrix column associated with variable x_j, it can be split into two (or more) sparser columns A_j^1 and A_j^2 that, when combined, intersect the same rows as A_j. By defining A_j^1 and A_j^2 so that $A_j = A_j^1 + A_j^2$, and their non-zero indices do not intersect, a single dense column can be replaced by two sparser ones. In the model formulation, $A_j x_j$ is replaced by $A_j^1 x_j^1 + A_j^2 x_j^2$. The model also requires an additional constraint $x_j^1 - x_j^2 = 0$. The value of x_j in the final solution is x_j^1 (or x_j^2).

Column splitting increases the problem size to improve performance. However, this somewhat counterintuitive approach can be effective because the reduction in run time associated with the dense columns exceeds the increase in run time associated with additional variables and constraints. More generally, if an increase in problem size removes a performance problem in an algorithm without creating a new bottleneck, it may improve overall algorithm performance.

Table 3 summarizes problems and suggestions for resolving them as addressed in Section 3. These suggestions assume that the practitioner has formulated the model in question with the correct fidelity, i.e., that the model cannot be reduced without compromising the value of its solution. When the model size becomes potentially problematic, as illustrated in Iteration Logs #2 and #14, the practitioner should also consider whether solving a less refined, smaller model would answer his needs.

When neither the tactics in Table 3 nor any other tuning of the simplex method or barrier algorithms yields satisfactory performance, the practitioner may also consider more advanced variants of the simplex method. Such variants frequently solve a sequence of easier, more tractable LPs, resulting in an optimal solution to the more difficult LP of interest. Some variants, such as sifting (also called the SPRINT technique by its originator, John Forrest [38]), work on parts of the LP of interest, solving a sequence of LPs in which the solution of the previous one defines problem modifications for the next one. In particular, sifting works well on LPs in which the number of variables dramatically exceeds the number of constraints. It starts with an LP consisting of all constraints but a manageable subset of the variables. After solving

Table 3
LP Performance issues and their suggested resolution.

LP performance issue	Suggested resolution
Numerical instability	• Calculate and input model data in double precision • Eliminate nearly-redundant rows and/or columns of *A a priori* • Avoid mixtures of large and small numbers: (i) Be suspicious of κ between 10^{10} and 10^{14}; (ii) Avoid data leading to κ greater than 10^{14} • Use alternate scaling (in the model formulation or optimizer settings) • Increase the Markowitz threshold • Employ the numerical emphasis parameter (if available)
Lack of objective function improvement under degeneracy	• Try all other algorithms (and variants) • Perturb data either *a priori* or using algorithmic settings
Primal degeneracy	• Use either dual simplex or interior point on primal problem
Dual degeneracy	• Employ either primal simplex or interior point on primal problem
Both primal and dual degeneracy	• Execute interior point on primal or dual problem
Excessive time per iteration	• Try all other algorithms (and variants) • Use algorithmic settings to conserve memory or purchase more externally • Try less expensive pricing settings if using simplex algorithms
Excessive simplex algorithm iterations	• Try Steepest edge or Devex variable selection
Multiple bound shift removals or significant infeasibilities after removing shifts	• Reduce feasibility and optimality tolerances
Barrier algorithm iterations with little or no progress	• Increase barrier convergence tolerance in order to initiate crossover earlier
Too much time in crossover	• Reduce barrier convergence tolerance in order to provide a better starting point for crossover

this LP subproblem, it uses the optimal dual variables to compute reduced costs that identify potential variables to add. The basis from the previous LP is typically primal feasible, speeding up the next optimization. Through careful management of added and removed variables from the sequence of LPs it solves, sifting can frequently solve the LP of interest to optimality without ever having to represent it in memory in its entirety. Sifting can also work well on models for which the number of constraints dramatically exceeds the number of variables by applying sifting to the dual LP. Other variants are more complex, maintaining two or more distinct subproblems, and using the solution of one to modify the other. The most frequently used methods are Dantzig-Wolfe Decomposition and Benders' Decomposition. These methods both maintain a master problem and solve a separate subproblem to generate modifications to the master problem. Both the master and subproblem are typically much easier to solve than the original LP of interest. Repeatedly optimizing the master and subproblem ultimately yields an optimal solution to the original problem. The Dantzig-Wolfe Decomposition is column-based, using the subproblem to generate additional columns for the master problem, while Benders' Decomposition is row-based, solving the subproblem to generate additional rows for the master problem. It can be shown that Benders' Decomposition applied to the primal representation of an LP is equivalent to applying Dantzig-Wolfe Decomposition to the dual LP. The details and application of these and other decomposition methods are beyond the scope of this paper. Dantzig and Thapa [39] provide more information about these approaches.

4. Conclusions

We summarize in this paper commonly employed linear programming algorithms and use this summary as a basis from which we present likely algorithmic troubles and associated avenues for their resolution. While optimizers and hardware will continue to advance in their capabilities of handling hard linear programs, practitioners will take advantage of corresponding improved performance to further refine their models. The guidelines we present are useful, regardless of the anticipated advances in hardware and software. Practitioners can implement many of these guidelines without expert knowledge of the underlying theory of linear programming, thereby enabling them to solve larger and more detailed models with existing technology.

Acknowledgments

Dr. Klotz wishes to acknowledge all of the CPLEX practitioners over the years, many of whom have provided the wide variety of models that revealed the guidelines described in this paper. He also wishes to thank the past and present CPLEX development, support, and sales and marketing teams who have contributed to the evolution of the product. Professor Newman wishes to thank the students in her first advanced linear programming class at the Colorado School of Mines for their helpful comments; she also wishes to thank her colleagues Professor Josef Kallrath (BASF-AG, Ludwigshafen, Germany) and Jennifer Rausch (Jeppeson, Englewood, Colorado) for helpful comments on an earlier draft. Both authors thank an anonymous referee for his helpful comments that lead to the improvement of the paper.

Both authors also wish to remember Lloyd Clarke (February 14, 1964–September 20, 2007). His departure from the CPLEX team had consequences that extended beyond the loss of an important employee and colleague.

References

[1] G. Dantzig, Linear Programming and Extensions, Princeton University Press, 1963.
[2] IBM, ILOG CPLEX. Incline Village, NV, 2012.
[3] Gurobi, 2012. Gurobi Optimizer. Houston, TX.
[4] MOPS, MOPS, Paderborn, Germany, 2012.
[5] MOSEK, MOSEK Optimization Software, Copenhagen, Denmark, 2012.
[6] FICO, Xpress-MP Optimization Suite. Minneapolis, MN, 2012.
[7] V. Chvátal, Linear Programming, W. H. Freeman, 1983.
[8] G. Dantzig, M. Thapa, Linear Programming 1: Introduction, Springer, 1997.
[9] R. Rardin, Optimization in Operations Research, Prentice Hall, 1998, (Chapter 6).
[10] W. Winston, Operations Research: Applications and Algorithms, Brooks/Cole, Thompson Learning, 2004.
[11] M. Bazaraa, J. Jarvis, H. Sherali, Linear Programming and Network Flows, John Wiley & Sons, Inc., 2005.
[12] G. Brown, R. Rosenthal, Optimization tradecraft: Hard-won insights from real-world decision support, Interfaces 38 (5) (2008) 356–366.
[13] C. Lemke, The dual method of solving the linear programming problem, Naval Research Logistics Quarterly 1 (1) (1954) 36–47.
[14] I. Duff, A. Erisman, J. Reid, Direct Methods for Sparse Matrices, Clarendon Press Oxford, 1986.
[15] J. Nazareth, Computer Solution of Linear Programs, Oxford University Press, 1987.
[16] B. Fourer, Notes on the dual simplex method, 1994, http://users.iems.northwestern.edu/~4er/WRITINGS/dual.pdf.
[17] D. Bertsimas, J. Tsitsiklis, Introduction to Linear Optimization, Prentice Hall, 1997, (Chapter 4).
[18] I. Dikin, Iterative solution of problems of linear and quadratic programming, Soviet Mathematics Doklady 8 (1967) 674–675.
[19] A. Fiacco, G. McCormick, Nonlinear Programming: Sequential Unconstrained Minimization Techniques, Wiley, 1968.
[20] L. Khachian, A polynomial algorithm for linear programming, Soviet Mathematics Doklady 20 (1979) 191–194.
[21] N. Karmarkar, A new polynomial-time algorithm for linear programming, Combinatorica 4 (4) (1984) 373–395.
[22] P. Gill, W. Murray, M. Saunders, J. Tomlin, M. Wright, On projected Newton barrier methods for linear programming and an equivalence to Karmarkar's projective method, Mathematical Programming 36 (2) (1986) 183–209.
[23] I. Lustig, R. Marsten, D. Shanno, Interior-point methods for linear programming: computational state of the art, ORSA Journal on Computing 6 (1) (1994) 1–14.
[24] I. Lustig, R. Marsten, M. Saltzman, R. Subramanian, D. Shanno, Interior-point methods for linear programming: just call Newton, Lagrange, and Fiacco and McCormick! Interfaces 20 (4) (1990) 105–116.
[25] S. Mehrotra, On the implementation of a primal–dual interior point method, SIAM Journal on Optimization 2 (4) (1992) 575–601.
[26] N. Megiddo, On finding primal- and dual- optimal bases, ORSA Journal on Computing 3 (1) (1991) 63–65.
[27] S. Wright, Primal-Dual Interior-Point Methods, SIAM, 1997.
[28] N. Higham, Accuracy and Stability of Numerical Algorithms, SIAM, 1996.
[29] Cybernet Systems Co., Maple, 2012, http://www.maplesoft.com/.
[30] Wolfram, Mathematica, 2012, http://www.wolfram.com/mathematica/.
[31] D. Applegate, W. Cook, S. Dash, QSopt, 2005, http://www.isye.gatech.edu/~wcook/qsopt/.
[32] D. Goldfarb, J. Reid, A practical steepest-edge simplex algorithm, Mathematical Programming 12 (1) (1977) 361–371.
[33] H. Greenberg, J. Kalan, An exact update for Harris' TREAD, Mathematical Programming Study 4 (1) (1975) 26–29.
[34] P. Harris, Pivot selection methods in the Devex LP code, Mathematical Programming Study 4 (1) (1975) 30–57.
[35] D. Goldfarb, J. Forrest, Steepest-edge simplex algorithms for linear programming, Mathematical Programming 57 (1) (1992) 341–374.
[36] I. Lustig, J. Mulvey, T. Carpenter, Formulating two-stage stochastic programs for interior point methods, Operations Research 39 (5) (1991) 757–770.
[37] R. Vanderbei, Splitting dense columns in sparse linear systems, Linear Algebra and its Applications 152 (1) (1991) 107–117.
[38] J. Forrest, Mathematical programming with a library of optimization routines, Presentation at 1989 ORSA/TIMS Joint National Meeting, 1989.
[39] G. Dantzig, M. Thapa, Linear Programming 2: Theory and Extensions, Springer, 2003.

Surveys in Operations Research and Management Science 18 (2013) 18–32

Contents lists available at ScienceDirect

Surveys in Operations Research and Management Science

journal homepage: www.elsevier.com/locate/sorms

Review

Practical guidelines for solving difficult mixed integer linear programs

Ed Klotz [a], Alexandra M. Newman [b,*]

[a] IBM, Incline Village, NV 89451, United States

[b] Division of Economics and Business, Colorado School of Mines, Golden, CO 80401, United States

ARTICLE INFO

Article history:
Received 5 August 2012
Received in revised form
3 December 2012
Accepted 14 December 2012

ABSTRACT

Even with state-of-the-art hardware and software, mixed integer programs can require hours, or even days, of run time and are not guaranteed to yield an optimal (or near-optimal, or any!) solution. In this paper, we present suggestions for appropriate use of state-of-the-art optimizers and guidelines for careful formulation, both of which can vastly improve performance.

Contents

"*Problems worthy of attack prove their worth by hitting back.*"
Piet Hein, Grooks 1966

"*Everybody has a plan until he gets hit in the mouth.*"
Mike Tyson

1. Introduction

Operations research practitioners have been formulating and solving integer programs since the 1950s. As computer hardware has improved [1], practitioners have taken the liberty to formulate increasingly detailed and complex problems, assuming that the corresponding instances can be solved. Indeed, commercially available optimizers such as CPLEX [2], Gurobi [3], MOPS [4], Mosek [5], and Xpress-MP [6] can solve many practical large-scale integer programs effectively. However, even if these "real-world" problem instances are solvable in an acceptable amount of time (seconds, minutes or hours, depending on the application), other instances require days or weeks of solution time. Although not a guarantee of tractability, carefully formulating the model and tuning standard integer programming algorithms often result in significantly faster solve times, in some cases, admitting a feasible or near-optimal solution which could otherwise elude the practitioner.

In this paper, we briefly introduce integer programs and their corresponding commonly used algorithm, show how to assess optimizer performance on such problems through the respective algorithmic output, and demonstrate methods for improving that performance through careful formulation and algorithmic parameter tuning. Specifically, there are many mathematically equivalent ways in which to express a model, and each optimizer has its own set of default algorithmic parameter settings. Choosing from these various model expressions and algorithmic settings can profoundly influence solution time. Although it is theoretically possible to try each combination of parameter settings, in practice, random experimentation would require vast amounts of time and would be unlikely to yield significant improvements. We therefore guide the reader to likely performance-enhancing parameter settings given fixed hardware, e.g., memory limits, and suggest

* Corresponding author.
E-mail addresses: klotz@us.ibm.com (E. Klotz), newman@mines.edu (A.M. Newman).

1876-7354/$ – see front matter © 2013 Elsevier Ltd. All rights reserved.
doi:10.1016/j.sorms.2012.12.001

methods for avoiding performance failures *a priori* through careful model formulation. All of the guidelines we present here apply to the model in its entirety. Many relaxation and decomposition methods, e.g., Lagrangian Relaxation, Benders' Decomposition, and Column Generation (Dantzig–Wolfe Decomposition), have successfully been used to make large problems more tractable by partitioning the model into subproblems and solving these iteratively. A description of these methods is beyond the scope of our paper; the practitioner should first consider attempting to improve algorithmic performance or tighten the existing model formulation, as these approaches are typically easier and less time consuming than reformulating the model and applying decomposition methods.

The reader should note that we assume basic familiarity with fundamental mathematics, such as matrix algebra, and with optimization, in particular, with linear programming and the concepts contained in Klotz and Newman [7]. We expect that the reader has formulated linear integer programs and has a conceptual understanding of how the corresponding problems can be solved. Furthermore, we present an algebraic, rather than a geometric, tutorial, i.e., a tutorial based on the mathematical structure of the problem and corresponding numerical algorithmic output, rather than based on graphical analysis. The interested reader can refer to basic texts such as Rardin [8] and Winston [9] for more detailed introductions to mathematical programming, including geometric interpretations.

We have attempted to write this paper to appeal to a diverse audience. Readers with limited mathematical programming experience who infrequently use optimization software and do not wish to learn the details regarding how the underlying algorithms relate to model formulations can still benefit from this paper by learning how to identify sources of slow performance based on optimizer output. This identification will allow them to use the tables in the paper that list potential performance problems and parameter settings that address them. More experienced practitioners who are interested in the way in which the optimizer algorithm relates to the model formulation will gain insight into new techniques for improving model formulations, including those different from the ones discussed in this paper. While intended primarily for practitioners seeking performance enhancements to practical models, theoretical researchers may still benefit. The same guidelines that can help tighten specific practical models can also help in the development of the theory associated with fundamental algorithmic improvements in integer programming, e.g., new cuts and new techniques for preprocessing.

The remainder of the paper is organized as follows: In Section 2, we introduce integer programs, the branch-and-bound algorithm, and its variants. Section 3 provides suggestions for successful algorithm performance. Section 4 presents guidelines for and examples of, tight formulations of integer programs that lead to faster solution times. Section 5 concludes the paper with a summary. Section 2, with the exception of the tables, may be omitted without loss of continuity for the practitioner interested only in formulation and algorithmic parameter tuning without detailed descriptions of the algorithms themselves. To illustrate the concepts we present in this paper, we show output logs resulting from having run a commercial optimizer on a standard desktop machine. Unless otherwise noted, this optimizer is CPLEX 12.2.0.2, and the machine possesses four single-core 3.0 GHz Xeon chips and 8 GB of memory.

2. Fundamentals

Consider the following system in which C is a set of indices on our variables x such that $x_j,\ j \in C$ are nonnegative, continuous variables, and I is a set of indices on the variables x such that $x_j,\ j \in I$ are nonnegative, integer variables. Correspondingly, c_C and A_C are the objective function and left-hand-side constraint coefficients, respectively, on the nonnegative, continuous variables, and c_I and A_I are the objective function and left-hand-side constraint coefficients, respectively, on the nonnegative, integer variables. For the constraint set, the right-hand-side constants, b, are given as an $m \times 1$ column vector.

$$(P_{MIP}):\ \min\ c_C^T x_C + c_I^T x_I$$
$$\text{subject to } A_C x_C + A_I x_I = b$$
$$x_C, x_I \geq 0, \quad x_I \text{ integer.}$$

Three noteworthy special cases of this standard mixed integer program are (i) the case in which x_I is binary, (ii) the case in which c_C, A_C, and x_C do not exist and x_I is general integer, and (iii) the case in which c_C, A_C, and x_C do not exist and x_I is binary. Note that (iii) is a special case of (i) and (ii). We refer to the first case as a mixed binary program, the second case as a pure integer program, and the third case as a binary program. These cases can benefit from procedures such as probing on binary variables [10], or even specialized algorithms. For example, binary programs lend themselves to some established techniques in the literature that do not exist if the algorithm is executed on an integer program. These techniques are included in most standard branch-and-bound optimizers; however, some features that are specific to binary-only models, e.g., the additive algorithm of Balas [11], can be lacking.

Branch-and-bound uses intelligent enumeration to arrive at an optimal solution for a (mixed) integer program or any special case thereof. This involves construction of a search tree. Each node in the tree consists of the original constraints in (P_{MIP}), along with some additional constraints on the bounds of the integer variables, x_I, to induce those variables to assume integer values. Thus, each node is also a mixed integer program (MIP). At each node of the branch-and-bound tree, the algorithm solves a linear programming relaxation of the restricted problem, i.e., the MIP with all its variables relaxed to be continuous.

The *root node* at the top of the tree is (P_{MIP}) with the variables x_I relaxed to assume continuous values. Branch-and-bound begins by solving this problem. If the root node linear program (LP) is infeasible, then the original problem (which is more restricted than its linear programming relaxation) is also infeasible, and the algorithm terminates with no feasible solution. Similarly, if the optimal solution to the root node LP has no integer restricted variables with fractional values, then the solution is optimal for (P_{MIP}) as well. The most likely case is that the algorithm produces an optimal solution for the relaxation with some of the integer-restricted variables assuming fractional values. In this case, such a variable, $x_j = f$, is chosen and *branched on*, i.e., two *subproblems* are created — one with a restriction that $x_j \leq \lfloor f \rfloor$ and the other with a restriction that $x_j \geq \lceil f \rceil$. These subproblems are successively solved, which results in one of the following three outcomes:

Subproblem solution outcomes (for a minimization problem)

- (i) *The subproblem is optimal with all variables in I assuming integer values*. In this case, the algorithm can update its best integer feasible solution; this update tightens the upper bound on the optimal objective value. Because the algorithm only seeks a single optimal solution, no additional branches are created from this node; examining additional branches cannot yield a better integer feasible solution. Therefore, the node is *fathomed* or *pruned*.
- (ii) *The subproblem is infeasible*. In this case, no additional branching can restore feasibility. As in (i), the node is fathomed.
- (iii) *The subproblem has an optimal solution, but with some of the integer-restricted variables in I assuming fractional values*. There are two cases:

- ⋆ a. The objective function value is dominated by the objective of the best integer feasible solution. In other words, the optimal node LP objective is no better than the previously established upper bound on the optimal objective for (P_{MIP}). In this case, no additional branching can improve the objective function value of the node, and, as in (i), the node is fathomed.
- ⋆ b. The objective function value is not dominated by that of the best integer feasible solution. The algorithm then *processes* the node in that it chooses a fractional $x_{j'} = f'; \quad j' \in I$ to branch on by creating two *child* nodes and their associated subproblems — one with a restriction that $x_{j'} \leq \lfloor f' \rfloor$ and the other with a restriction that $x_{j'} \geq \lceil f' \rceil$. These restrictions are imposed on the subproblem in addition to any others from previous branches in the same chain stemming from the root; each of these child subproblems is subsequently solved. Note that while most implementations of the algorithm choose a single integer variable from which to create two child nodes, the algorithm's convergence only requires that the branching divides the feasible region of the current node in a mutually exclusive manner. Thus, branching on groups of variables or expressions of variables is also possible.

Due to the exponential growth in the size of such a tree, exhaustive enumeration would quickly become hopelessly computationally expensive for MIPs with even dozens of variables. The effectiveness of the branch-and-bound algorithm depends on its ability to prune nodes. Effective pruning relies on the fundamental property that the objective function value of each child node is either the same as or worse than that of the parent node (both for the MIP at the node and the associated LP relaxation). This property holds because every child node consists of the MIP in the parent node plus an additional constraint (typically, the bound constraint on the branching variable).

As the algorithm proceeds, it maintains the incumbent integer feasible solution with the best objective function determined thus far in the search. The algorithm performs updates as given in (i) of Subproblem Solution Outcomes. The updated incumbent objective value provides an upper bound on the optimal objective value. A better incumbent increases the number of nodes that can be pruned in case (iii), part (a) by more easily dominating objective function values elsewhere in the tree.

In addition, the algorithm maintains an updated lower bound on the optimal objective for (P_{MIP}). The objective of the root node LP establishes a lower bound on the optimal objective because its feasible region contains all integer feasible solutions to (P_{MIP}). As the algorithm proceeds, it dynamically updates the lower bound by making use of the property that the child node objectives are no better than those of their parent. Because a better integer solution can only be produced by the children of the currently unexplored nodes, this property implies that the optimal objective value for (P_{MIP}) can be no better than the best unexplored node LP objective value. As the algorithm continues to process nodes, the minimum LP objective of the unexplored nodes can dynamically increase, improving the lower bound. When the lower bound meets the upper bound, the algorithm terminates with an optimal solution. Furthermore, once an incumbent has been established, the algorithm uses the difference between the upper bound and lower bound to measure the quality of the solution relative to optimality. Thus, on difficult models with limited computation time available, practitioners can configure the algorithm to stop as soon as it has an integer feasible solution within a specified percentage of optimality. Note that most other approaches to solving integer programs (e.g., tabu search, genetic algorithms) lack any sort of bound, although it may be possible to derive one from the model instance. However, even if it is possible to derive a bound, it is likely to be weak, and it probably remains static. Note that in the case of a maximization problem, the best integer solution provides a lower bound on the objective function value and the objective of the root node LP establishes an upper bound on the optimal objective; the previous discussion holds, but with this reversal in bounds. Unless otherwise noted, our examples are minimization problems, as given by our standard form in (P_{MIP}).

Fig. 1 provides a tree used to solve a hypothetical integer program of the form (P_{MIP}) with the branch-and-bound algorithm. Only the relevant subset of solution values is given at each node. The numbers in parentheses outside the nodes denote the order in which the nodes are processed, or examined. The inequalities on the arcs indicate the bound constraint placed on an integer-restricted variable in the original problem that possesses a fractional value in a subproblem.

Node (1) is the root node. Its objective function value provides a lower bound on the minimization problem. Suppose x_1, an integer-restricted variable in the original problem, possesses a fractional value (3.5) at the root node solve. To preclude this fractional value from recurring in any subsequent child node solve, we create two subproblems, one with the restriction that $x_1 \leq 3$, i.e., $x_1 \leq \lfloor 3.5 \rfloor$, and the other with the restriction that $x_1 \geq 4$, i.e., $x_1 \geq \lceil 3.5 \rceil$. This is a mutually exclusive and collectively exhaustive set of outcomes for x_1 (and, hence, the original MIP) given that x_1 is an integer-restricted variable in the original problem.

Node (2) is the child node that results from branching down on variable x_1 at node (1). Among possibly others, x_7 is an integer-restricted variable that assumes a fractional value when this subproblem at node (2) is solved; the solve consists of the root node problem and the additional restriction that $x_1 \leq 3$. Because of this fractional value, we create two subproblems emanating from node (2) in the same way in which we create them from node (1). The subproblem solve at node (4), i.e., the solve consisting of the root node subproblem plus the two additional restrictions that $x_1 \leq 3$ and $x_7 \leq 2$, results in an integer solution. At this point, we can update the upper bound. That is, the optimal solution for this problem, an instance of (P_{MIP}), can never yield an objective worse than that of the best feasible solution obtained in the tree.

At any point in the tree, nodes that require additional branching are considered active, or unexplored. Nodes (6) and (11) remain unexplored. Additional processing has led to pruned nodes (4), (7), and (9), either because the subproblem solve was infeasible, e.g., node (9), or because the objective function value was worse than that of node (4), regardless of whether or not the resulting solution was integer. As the algorithm progresses, it establishes an incumbent solution at node (10). Because nodes (6) and (11) remain unexplored, improvement on the current incumbent can only come from the solutions of the subproblems at nodes (6) and (11) or their descendants. The descendants have an objective function value no better than that of either of these two nodes; therefore, the optimal solution objective is bounded by the minimum of the optimal LP objectives of nodes (6) and (11). Without loss of generality, assume node (11) possesses the lesser objective. That objective value then provides a lower bound on the optimal objective for (P_{MIP}). We can continue searching through the tree in this fashion, updating lower and upper bounds, until either the gap is acceptably small, or until all the nodes have been processed.

The previous description of the branch-and-bound algorithm focuses on its fundamental steps. Advances in the last 20 years have extended the algorithm from branch and bound to branch and cut. Branch and cut, the current choice of most integer programming solvers, follows the same steps as branch and bound, but it also can add cuts. Cuts consist of constraints involving linear expressions of one or more variables that are added at the nodes to further

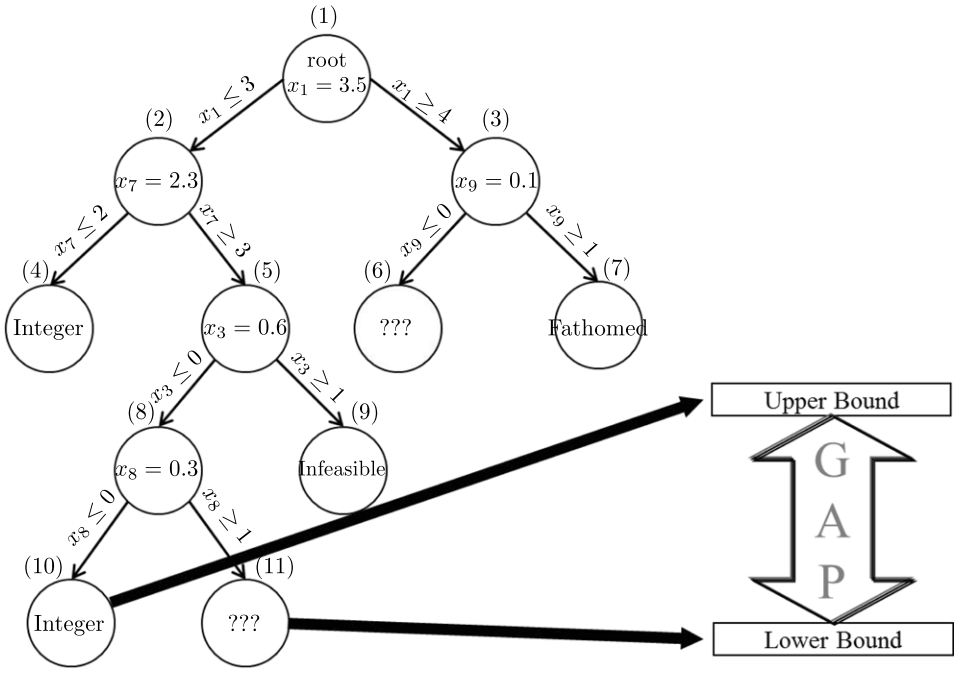


Fig. 1. Branch-and-bound algorithm.

improve performance. As long as these cuts do not remove any integer feasible solutions, their addition does not compromise the correctness of the algorithm. If done judiciously, the addition of such cuts can yield significant performance improvements.

3. Guidelines for successful algorithm performance

There are four common reasons that integer programs can require a significant amount of solution time:

- (i) There is lack of node throughput due to troublesome linear programming node solves.
- (ii) There is lack of progress in the best integer solution, i.e., the upper bound.
- (iii) There is lack of progress in the best lower bound.
- (iv) There is insufficient node throughput due to numerical instability in the problem data or excessive memory usage.

By examining the output of the branch-and-bound algorithm, one can often identify the cause(s) of the performance problem. Note that integer programs can exhibit dramatic variations in run time due to seemingly inconsequential changes to a problem instance. Specifically, differences such as reordering matrix rows or columns, or solving a model with the same optimizer, but on a different operating system, only affect the computations at very low-order decimal places. However, because most linear programming problems drawn from practical sources have numerous alternate optimal basic solutions, these slight changes frequently suffice to alter the path taken by the primal or dual simplex method. The fractional variables eligible for branching are basic in the optimal node LP solution. Therefore, alternate optimal bases can result in different branching variable selections. Different branching selections, in turn, can cause significant performance variation if the model formulation or optimizer features are not sufficiently robust to consistently solve the model quickly. This notion of performance variability in integer programs is discussed in more detail in Danna [12] and Koch et al. [13]. However, regardless of whether an integer program is consistently or only occasionally difficult to solve, the guidelines described in this section can help address the performance problem. We now discuss each potential performance bottleneck and suggest an associated remedy.

3.1. Lack of node throughput due to troublesome linear programming node solves

Because processing each node in the branch-and-bound tree requires the solution of a linear program, the choice of a linear programming algorithm can profoundly influence performance. An interior point method may be used for the root node solve; it is less frequently used than the simplex method at the child nodes because it lacks a basis and hence, the ability to start with an initial solution, which is important when processing tens or hundreds of thousands of nodes. However, conducting different runs in which the practitioner invokes the primal or the dual simplex method at the child nodes is a good idea. Consider the following two node logs, the former corresponding to solving the root and child node linear programs with the dual simplex method and the latter with the primal simplex method.

The iteration count for the root node solve shown in *Node Log* #1 that occurred without any advanced start information indicates 5278 iterations. Computing the average iteration count across all node LP solves, there are 11 solves (10 nodes, and 1 extra solve for cut generation at node 0) and 73,714 iterations, which were performed in a total of 177 seconds. The summary output in gray indicates in parentheses that one unexplored node remains. So, the average solution time per node is approximately 17 seconds, and the average number of iterations per node is about 6701. In *Node Log* #2, the solution time is 54 seconds, at which point the algorithm has performed 11 solves, and the iteration count is 23,891. The average number of iterations per node is about 2172. In *Node Log* #1, the 10 child node LPs require more iterations, 6844, on average, than the root node LP (which requires 5278), despite the advanced basis at the child node solves that was absent at the root node solve. Any time this is true, or even when the average node LP iteration count is more than 30%–50% of the root node iteration count, an opportunity for improving node LP solve times

Node Log #1: Node Linear Programs Solved with Dual Simplex

```
        Nodes                                          Cuts/          ItCnt
Node    Left    Objective       IInf    Best Integer   Best Node
0       0       -89.0000        6                      -89.0000       5278
0       0       -89.0000        6                      Fract: 4       12799
0       2       -89.0000        6                      -89.0000       12799
1       1       infeasible                             -89.0000       20767
2       2       -89.0000        5                      -89.0000       27275
3       1       infeasible                             -89.0000       32502
...
8       2       -89.0000        8                      -89.0000       65717
9       1       infeasible                             -89.0000       73714
...
Solution time = 177.33 sec.  Iterations = 73714  Nodes = 10 (1)
```

Node Log #2: Node Linear Programs Solved with Primal Simplex

```
        Nodes                                          Cuts/          ItCnt
Node    Left    Objective       IInf    Best Integer   Best Node
0       0       -89.0000        5                      -89.0000       6603
0       0       -89.0000        5                      Fract: 5       7120
0       2       -89.0000        5                      -89.0000       7120
1       1       infeasible                             -89.0000       9621
2       2       -89.0000        5                      -89.0000       10616
3       1       infeasible                             -89.0000       12963
...
8       2       -89.0000        8                      -89.0000       21522
9       1       infeasible                             -89.0000       23891
...
Solution time = 54.37 sec.  Iterations = 23891  Nodes = 10 (1)
```

exists by changing algorithms or algorithmic settings. In *Node Log* #2, the 10 child node LPs require 1729 iterations, on average, which is much fewer than those required by the root node solve, which requires 6603 (solving the LP from scratch). Hence, switching from the dual simplex method in *Node Log* #1 to the primal simplex method in *Node Log* #2 increases throughput, i.e., decreases the average number of iterations required to solve a subproblem in the branch-and-bound tree.

The different linear programming algorithms can also benefit by tuning the appropriate optimizer parameters. See Klotz and Newman [7] for a detailed discussion of this topic.

3.2. Lack of progress in the best integer solution

An integer programming algorithm may struggle to obtain good feasible solutions. *Node Log* #3 illustrates a best integer solution found before node 300 of the solve that has not improved by node 7800 of the same solve.

Many state-of-the-art optimizers have built-in heuristics to determine initial and improved integer solutions. However, it is always valuable for the practitioner to supply the algorithm with an initial solution, no matter how obvious it may appear to a human. Such a solution may provide a better starting point than what the algorithm can derive on its own, and algorithmic heuristics may perform better in the presence of an initial solution, regardless of the quality of its objective function value. In addition, the faster progress in the cutoff value associated with the best integer solution may enable the optimizer features such as probing to fix additional variables, further improving performance. Common tactics to find such starting solutions include the following:

- Provide an obvious solution based on specific knowledge of the model. For example, models with integer penalty variables may benefit from a starting solution with a significant number (or even all) of the penalty variables set to non-zero values.
- Solve a related, auxiliary problem to obtain a solution (e.g., via the Feasopt method in CPLEX, which looks for feasible solutions by minimizing infeasibilities), provided that the gain from the starting solution exceeds the auxiliary solve time.
- Use the solution from a previous solve for the next solve when solving a sequence of models.

To see the advantages of providing a starting point, compare *Node Log* #5 with *Node Log* #4. Log #4 shows that CPLEX with default settings takes about 1589 seconds to find a first feasible solution, with an associated gap of 4.18%. Log #5 illustrates the results obtained by solving a sequence of five faster optimizations (see Lambert et al. [14] for details) to obtain a starting solution with a gap of 2.23%. The total computation time to obtain the starting solution is 623 seconds. So, the time to obtain the first solution is faster by providing an initial feasible solution, and if we let the algorithm with the initial solution run for an additional $1589 - 623 = 966$ seconds, the gap for the instance with the initial solution improves to 1.53%.

In the absence of a readily identifiable initial solution, various branching strategies can aid in obtaining initial and subsequent solutions. These branching strategies may be based purely on the algebraic structure of the model. For example, by using depth-first search, the branch-and-bound algorithm never defers processing a node until it has been pruned. This strategy helps find integer feasible solutions sooner, although it potentially slows progress in the best bound. (Recall, the best lower bound for a

Node Log #3: Lack of Progress in Best Integer Solution

```
       Nodes                                          Cuts/        ItCnt     Gap
Node   Left   Objective   IInf  Best Integer          Best Node
...
300     229     22.6667     40   31.0000              22.0000      4433      29.03%
400     309     cutoff           31.0000              22.3333      5196      27.96%
500     387     26.5000     31   31.0000              23.6667      6164      26.88%
...
7800    5260    28.5000     23   31.0000              25.6667      55739     17.20%
```

Node Log #4: No initial practitioner-supplied solution

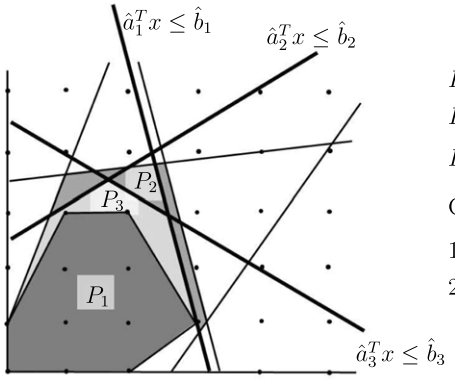
```
Root relaxation solution time =  131.45 sec.

        Nodes                                             Cuts/
   Node  Left     Objective  IInf  Best Integer     Best Node    ItCnt     Gap

      0     0   1.09590e+07  2424                1.09590e+07   108111
      0     0   1.09570e+07  2531                   Cuts: 4    108510
      0     0   1.09405e+07  2476                   Cuts: 2    109208
Heuristic still looking.
Heuristic still looking.
Heuristic still looking.
Heuristic still looking.
Heuristic still looking.
      0     2   1.09405e+07  2476                1.09405e+07   109208
Elapsed real time = 384.09 sec. (tree size =  0.01 MB)
      1     3   1.08913e+07  2488                1.09405e+07   109673
      2     4   1.09261e+07  2326                1.09405e+07   109977
...
   1776  1208   1.05645e+07    27                1.09164e+07   474242
   1814  1246   1.05588e+07    31                1.09164e+07   478648
   1847  1277   1.05554e+07   225                1.09164e+07   484687
*  1880+ 1300                        1.04780e+07  1.09164e+07   491469    4.18%
   1880  1302   1.05474e+07   228   1.04780e+07  1.09164e+07   491469    4.18%
Elapsed real time =  1589.38 sec.  (tree size = 63.86 MB)
```

Node Log #5: An initial solution supplied by the practitioner

```
Root relaxation solution time =   93.92 sec.

        Nodes                                             Cuts/
   Node  Left     Objective  IInf  Best Integer     Best Node    ItCnt     Gap

*     0+    0                        1.07197e+07                108111     ---
      0     0   1.09590e+07  2424   1.07197e+07  1.09590e+07   108111    2.23%
      0     0   1.09570e+07  2531   1.07197e+07     Cuts: 4    108538    2.21%
...

    485   433   1.09075e+07  2398   1.07197e+07  1.08840e+07   244077    1.53%
    487   434   1.08237e+07  2303   1.07197e+07  1.08840e+07   244350    1.53%
    497   439   1.08637e+07  1638   1.07197e+07  1.08840e+07   245391    1.53%
Elapsed real time =  750.11 sec.  (tree size = 32.61 MB)
    501   443   1.08503e+07  1561   1.07197e+07  1.08840e+07   245895     1.53%
...
Elapsed real time = 984.03 sec. (tree size = 33.00 MB)
   1263   674   1.08590e+07  2574   1.07197e+07  1.08840e+07   314814    1.53%
```

Node Log #6: Progress in Best Integer Solution but not in the Best Bound

	Node	Nodes Left	Objective	IInf	Best Integer	Cuts/ Best Node	ItCnt	Gap
	300	296	2018.00	27	3780.00	560.00	3703	85.19%
*	300+	296		0	2626.00	560.00	3703	78.67%
*	393	368		0	2590.00	560.00	4405	78.38%
	400	372	560.00	291	2590.00	560.00	4553	78.38%
	500	472	810.00	175	2590.00	560.00	5747	78.38%
...								
*	7740+	5183		0	1710.00	560.00	66026	67.25%
	7800	5240	1544.00	110	1710.00	560.00	66279	67.25%
	7900	5325	944.00	176	1710.00	560.00	66801	67.25%
	8000	5424	1468.00	93	1710.00	560.00	67732	67.25%



$P_1 := \text{conv}\{x \in \mathbb{Z}^n : Ax \le b, x \ge 0\}$
$P_2 := \{x \in \mathbb{R}^n : Ax \le b, x \ge 0\}$
$P_3 := P_2 \cap \{x \in \mathbb{R}^n : \hat{A}x \le \hat{b}\}$

Cuts must satisfy

1) $\hat{a}_i^T x \le \hat{b}_i \quad \forall x \in P_1$ (validity)
2) $\exists\ x \in P_2 : \hat{a}_i^T x > \hat{b}_i$ (separation)

Fig. 2. Convex hull.

minimization problem is updated once all nodes with relaxation objective value equal to the lower bound have been processed.) In other cases, branching strategies may involve specific aspects of the model. For example, branching up, i.e., processing the subproblem associated with the greater bound as a restriction on its branch, in the presence of many set partitioning constraints $\left(\sum_i x_i = 1,\ x_i \text{ binary}\right)$ not only fixes the variable on the associated branch in the constraint to 1, but it also fixes all other variables in the constraint to a value of 0 in the children of the current node. By contrast, branching down does not yield the ability to fix any additional variables.

Improvements to the model formulation can also yield better feasible solutions faster. Differentiation in the data, e.g., by adding appropriate discounting factors to cost coefficients in the objective function, helps the algorithm distinguish between dominated and dominating solutions, which expedites the discovery of improving solutions.

3.3. Lack of progress in the best bound

The branch-and-bound depiction in Fig. 1 and the corresponding discussion illustrate how the algorithm maintains and updates a lower bound on the objective function value for the minimization integer program. (Note that this would correspond to an upper bound for a maximization problem.) The ability to update the best bound effectively depends on the best objective function value of all active subproblems, i.e., the associated LP objective function value of the nodes that have not been fathomed. If successive subproblems, i.e., subproblems corresponding to nodes lying deeper in the tree, do not possess significantly worse objective function values, the bound does not readily approach the true objective function value of the original integer program. Furthermore, the greater the number of active, i.e., unfathomed, nodes deeper in the tree, the smaller the chance of a tight bound, which always corresponds to the weakest (lowest, for a minimization problem) objective function value of any active node. These objective function values, and the associated bounds they generate, in turn, depend on the strength of the model formulation, i.e., the difference between the polyhedron associated with the LP relaxation of (P_{MIP}) and the polyhedron consisting of the convex hull of all integer feasible solutions to (P_{MIP}). Fig. 2 provides an illustration. The region P_1 represents the convex hull of all integer feasible solutions of the MIP, while P_2 represents the feasible region of the LP relaxation. Adding cuts yields the region P_3, which contains all integer solutions of the MIP, but contains only a subset of the fractional solutions feasible for P_2. *Node Log* #6 exemplifies progress in best integer solution but not in the best bound.

To strengthen the bound, i.e., to make its value closer to that of the optimal objective function value of the integer program, we can modify the integer program by adding special constraints. These constraints, or *cuts*, do not excise any integer solutions that are feasible in the unmodified integer program. A cut that does not remove any integer solution is *valid*. However, the cuts remove portions of the feasible region that contain fractional solutions. If the removed area contains the fractional solution resulting from the LP relaxation of the integer program, we say the cut is *useful* [8], or that the cut *separates* the fractional solution from the resulting LP relaxation feasible region. In this case, the cut improves the bound by increasing the original LP objective. There are various problem structures that lend themselves to different types of cuts. Thus, we have a general sense of cuts that could be useful. However, without the LP relaxation solution, it is difficult to say a priori which cuts are definitely useful.

Let us consider the following numerical example, in this case, for ease of illustration, a maximization problem:

$$(P^{EX}_{Binary}) \quad \max \; 3x_1 + 2x_2 + x_3 + 2x_4 + x_5 \tag{1}$$

$$\text{subject to } x_1 + x_2 \leq 1 \tag{2}$$

$$x_1 + x_3 \leq 1 \tag{3}$$

$$x_2 + x_3 \leq 1 \tag{4}$$

$$4x_3 + 3x_4 + 5x_5 \leq 10 \tag{5}$$

$$x_1 + 2x_4 \leq 2 \tag{6}$$

$$3x_2 + 4x_5 \leq 5 \tag{7}$$

$$x_i \text{ binary} \quad \forall i. \tag{8}$$

A *cover cut* based on the *knapsack constraint* of (P^{EX}_{Binary}), $4x_3 + 3x_4 + 5x_5 \leq 10$, is $x_3 + x_4 + x_5 \leq 2$. That is, at most two of the three variables can assume a value of 1 while maintaining feasibility of the knapsack constraint (5). Adding this cut is valid since it is satisfied by all integer solutions feasible for the constraint. It also separates the fractional solution $\left(x_1 = 0, x_2 = 0, x_3 = 1, x_4 = \frac{1}{3}, x_5 = 1\right)$ from the LP relaxation feasible region. Now consider the three *packing constraints*, (2)–(4): $x_1 + x_2 \leq 1, x_1 + x_3 \leq 1$, and $x_2 + x_3 \leq 1$. We can construct a *conflict graph* (see Fig. 3) for the whole model, with each vertex corresponding to a binary variable and each edge corresponding to a pair of variables, both of which cannot assume a value of 1 in any feasible solution. A clique is a set of vertices such that every two in the set are connected by an edge. At most one variable in a clique can equal 1. Hence, the vertices associated with x_1, x_2 and x_3 form a clique, and we derive the cut: $x_1 + x_2 + x_3 \leq 1$. In addition, constraints (6) and (7) generate the edges $\{1, 4\}$ and $\{2, 5\}$ in the conflict graph, revealing the cuts $x_1 + x_4 \leq 1$ and $x_2 + x_5 \leq 1$. One could interpret these cuts as either clique cuts from the conflict graph, or cover cuts derived directly from constraints (6) and (7). Note that not only does each of these clique cuts separate fractional solutions from the LP relaxation feasible region (as did the cover cut above), but they are also useful in that they remove the LP relaxation solution $\left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{3}{4}, \frac{7}{8}\right)$ from the feasible region.

The derivations of both clique and cover cuts rely on identifying a linear expression of variables that assumes an integral value in any integer feasible solution, then determining the integer upper (right-hand-side) limit on the expression. In the case of the cover cut for our example (P^{EX}_{Binary}), x_3, x_4 and x_5 form a cover, which establishes that $x_3+x_4+x_5 \geq 3$ is infeasible for any integer solution to the model. Therefore, $x_3 + x_4 + x_5 \leq 2$ is valid for any integer feasible solution to (P^{EX}_{Binary}). Similarly, the clique in the conflict graph identifies the integral expression $x_1+x_2+x_3$ and establishes that $x_1 + x_2 + x_3 \geq 2$ is infeasible for any integer solution to the model. Therefore, $x_1 + x_2 + x_3 \leq 1$ is valid for any integer feasible solution to (P^{EX}_{Binary}). This cut removes fractional solutions such as $\left(x_1 = \frac{1}{2}, x_2 = \frac{1}{2}, x_3 = \frac{1}{2}\right)$. Making use of fractional infeasibility relative to integer expressions is a useful technique for deriving additional cuts, and is a special case of disjunctive programming [15].

Another mechanism to generate additional cuts includes the examination of the *complementary system*, i.e., one in which a binary variable x_i is substituted with $1 - x_i$. Consider a constraint similar to the knapsack constraint, but with the inequality reversed: $\sum_i a_i x_i \geq b$ (with $a_i, b > 0$). Let $\bar{x}_i = 1 - x_i$. Multiplying the inequality on the knapsack-like constraint by -1 and adding $\sum_i a_i$ to both sides, we obtain: $\sum_i a_i - \sum_i a_i x_i \leq -b + \sum_i a_i$. Substituting the complementary variables yields: $\sum_i a_i \bar{x}_i \leq -b + \sum_i a_i$. Note that when the right hand side is negative, the original constraint is infeasible. Otherwise, this yields a knapsack constraint on $\bar{x}_i$ from which cuts can be derived. Cover cuts involving the $\bar{x}_i$ can then be translated into cuts involving the original x_i variables.

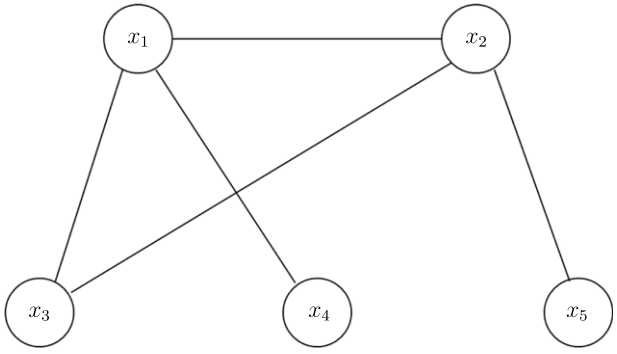


Fig. 3. Conflict Graph for numerical example (P^{EX}_{Binary}).

We summarize characteristics of these and other potentially helpful cuts in Table 1. A detailed discussion of each of these cuts is beyond the scope of this paper; see Achterberg [16] or Wolsey [17] for more details, as well as extensive additional references. State-of-the-art optimizers tend to implement cuts that are based on general polyhedral theory that applies to all integer programs, or on special structure that occurs on a sufficiently large percentage of practical models. Table 1 can help the practitioner distinguish cuts that a state-of-the-art optimizer is likely to implement from those that are specific to particular types of models, and are less likely to be implemented in a generic optimizer (and, hence, more likely to help performance if the practitioner uses his knowledge to derive them).

Adding cuts does not always help branch-and-bound performance. While it can remove integer infeasibilities, it also results in more constraints in each node LP. More constraints can increase the time required to solve these linear programs. Without a commensurate speed-up in solution time associated with processing fewer nodes, cuts may not be worth adding. Some optimizers have internal logic to automatically assess the trade-offs between adding cuts and node LP solve time. However, if the optimizer lacks such logic or fails to make a good decision, the practitioner may need to look at the branch-and-bound output in order to assess the relative increase in performance due to fewer examined nodes and the potential decrease in the rate at which the algorithm processes the nodes. In other cases, the computational effort required to derive the cuts needed to effectively solve the model may exceed the performance benefit they provide. Similar to node LP solve time and node throughput, a proper comparison of the reduction in solution time the cuts provide with the time spent calculating them may be necessary. (See Achterberg [16].)

Most optimizers offer parameter settings that can improve progress of the best node, either by strengthening the formulation or by enabling more node pruning. Features that are commonly available include:

- (i) *Best Bound node selection* By selecting the node with the minimal relaxation objective value, the algorithm updates the best node value faster. However, by considering node LP objective values while ignoring the number of integer infeasibilities, best bound node selection may cause the optimizer to find fewer integer feasible solutions. Therefore, best bound node selection

Table 1
Different types of cuts and their characteristics, where z is binary unless otherwise noted, and x is continuous.

Cut name	Mathematical description of cut	Structure of original MILP that generates the cut
Clique[b]	$\sum_i z_i \leq 1$	Packing constraints
Cover[b]	$\sum_i z_i \leq b$, b integer	Knapsack constraints
Disjunctive[a]	Constraint derived from an LP solution	$\sum_i a_i' x_i \geq b'$ or $\sum_i a_i'' x_i \geq b''$, x_i continuous or integer
Mixed Integer Rounding[a]	Use of floors and ceilings of coefficients and integrality of original variables	$a_C x_C + a_I x_I = b, x \geq 0$
Generalized Upper Bound[b]	$\sum_i x_i \leq b$, b integer	Knapsack constraints with precedence or packing
Implied Bound[b]	$x_i \leq \frac{b}{a_i}$	$\sum_i a_i x_i \leq bz, x \geq 0$
Gomory[a]	Mixed integer rounding applied to a simplex tableau row $\bar{a}$ associated with optimal node LP basis	$\bar{a}_C x_C + \bar{a}_{I/k} x_{I/k} + x_k = \bar{b}$, x_k integer, $x \geq 0$
Zero-half[a]	$\lambda^T A x \leq \lfloor \lambda^T b \rfloor$, $\lambda_i \in \{0, 1/2\}$	Constraints containing integer variables and coefficients
Flow Cover[b]	Linear combination of flow and binary variables involving a single node	Fixed charge network
Flow Path[b]	Linear combination of flow and binary variables involving a path of nodes	Fixed charge network
Multicommodity flow[b]	Linear combination of flow and binary variables involving nodes in a network cut	Fixed charge network

[a] Based on general polyhedral theory.
[b] Based on specific, commonly occurring problem structure.

is most likely to help performance on models in which the optimizer finds integer feasible solutions easily, but has trouble making sufficient progress in the best node.

- (ii) *Strong branching* By running a modest number of dual simplex iterations on multiple branching variable candidates at each node, the algorithm can exploit any infeasible branches to tighten additional variable bounds, resulting in a stronger formulation of the MIP at the node in question, and faster pruning of its descendants. Strong branching increases the computation at each node, so the performance improvement from the additional node pruning must compensate for the diminished rate of node throughput to make this a reasonable feature to employ.
- (iii) *Probing* By fixing a binary variable to a value of 0 or 1 and propagating this bound change to other variables through the intersecting constraints, the optimizer can often identify binary variables that can only assume one value in any feasible solution. For example, if fixing a binary variable to 0 establishes that (P_{MIP}) is infeasible, then the variable must be 1 in any integer feasible solution. Probing computation time primarily occurs as a preprocessing step before starting the branch-and-bound algorithm. Identifying binary variables to fix can tighten the formulation and improve node throughput by reducing the size of the problem. However, it can be computationally expensive, so the practitioner must compare the time spent performing the initial probing computations with the subsequent performance gains.
- (iv) *More aggressive levels of cut generation* Generating more cuts can further tighten the formulation. However, the practitioner must properly assess the trade-off between the tighter formulation and the potentially slower rate of node processing due to the additional constraints in the node LPs.

If alternate parameter settings are insufficient to yield progress in the best node, the following guidelines, while requiring more work, can help address this performance problem:

- (i) *Careful model formulation* It is sometimes possible to use alternate variable definitions. For example, in Bertsimas and Stock Patterson [18], the authors use variables to denote whether an aircraft (flight) has arrived at a sector in the airspace *by* time period t, and postulate that the variables represented in this manner "define connectivity constraints that are facets of the convex hull of solutions", which greatly improves the tractability of their model. Similarly, in a model designed to determine a net present value-maximizing schedule for extracting three-dimensional notional blocks of material in an open pit mine, we can define $x_{bt} = 1$ if block b is extracted *by* time period t, 0 otherwise, as opposed to the more intuitive $\hat{x}_{bt} = 1$ if block b is extracted *at* time period t, 0 otherwise [14]. The definitions in these two references result in models with significant differences in performance, as illustrated theoretically and empirically.
- (ii) *Careful use of elastic variables, i.e., variables that relax a constraint by allowing for violations (which are then penalized in the objective)* Adding elastic variables can result in MIPs that remove the infeasibilities on integer expressions essential to standard cut generation. This leads to a weaker model formulation in which most cut generation mechanisms are disabled. If the use of elastic variables is necessary, consider first minimizing the sum of the elastic variables, then optimizing the original objective while constraining the elastic variable values to their minimized values.

3.4. Data and memory problems

Because the optimizer solves linear programs at each node of the branch-and-bound tree, the practitioner must be careful to avoid the numerical performance issues described in Section 3 of Klotz and Newman [7]. Specifically, it is important to avoid large differences in orders of magnitude in data to preclude the introduction of unnecessary round-off error. Such differences of input values create round-off error in floating point calculations which makes it difficult for the algorithm to distinguish between this error and a legitimate value. If the algorithm makes the wrong distinction, it arrives at an incorrect solution. Integer programs may contain the construct "if $z = 0$, then $x = 0$. Otherwise, x can be arbitrarily large". Arbitrarily large values of x can be carelessly modeled with a numerical value designed to represent infinity (often referred to as "big M" in the literature). In reality, the value for this variable can be limited by other constraints in the problem; if so, we reduce its value, as in the following:

$$x - 100000000000z \leq 0 \tag{9}$$

$$0 \leq x \leq 5000; \quad z \text{ binary}. \tag{10}$$

In this case, we should use a coefficient of 5000 on z, which allows us to eliminate the explicit upper bound on x as well. In addition to improving the scaling of the constraint, this change to the numerical value enables the optimizer to better identify legitimate solutions to the conditions being modeled. For example, the unmodified constraint accepts values of $z = 10^{-8}$ and $x = 1000$ as an integer feasible solution. Most optimizers use an integrality tolerance and, by default, accept an integrality violation of this order of magnitude. Therefore, the big M coefficient on the original constraint enables the optimizer to accept a solution that, while feasible in a finite precision computing environment, does not satisfy the intended meaning of the constraint. See Camm et al. [19] for further discussion.

Branch-and-bound can be generalized to other logic, which is important because it removes the urge to use these numerically problematic "big M's" by allowing, for example, direct branching on an indicator constraint. The indicator formulation of (9) is $z = 0 \Rightarrow x \leq 0$. An indicator infeasibility that requires branching occurs when a node relaxation solution has $z = 0$ but $x > 0$. The indicator branches would be: $x \leq 0$ and $z = 1$. By contrast, large values in (9) or elsewhere in the model (whether truly infinite or some big M approximation) can result in a wide range of coefficients that can easily lead to numerical problems. So, using indicators eliminates these potentially large values from the matrix coefficients used to approximate an infinite value. For the case in which the large values impose meaningful limits in the model, the indicator formulation moves the coefficients from the matrix into the variable bounds, which improves the numerical characteristics of the model.

Indicator constraints also support more general conditions, e.g., $z = 0 \Rightarrow a^T x \leq b$. In this case, the indicator branches would be $a^T x \leq b$ and $z = 1$. However, relaxations of indicator constraints remove the constraint completely and can therefore be potentially weaker than their less numerically stable big M counterpart. As of this writing, recent improvements in indicator preprocessing in CPLEX have helped address this drawback.

Integer programs require at least as much memory as their linear programming equivalents. Running out of memory is therefore as frequent, if not more frequent, a problem when trying to solve integer programs, as opposed to linear programs. The same suggestions as those that appear in Section 3.3 of Klotz and Newman [7] apply.

Table 2 provides suggestions for the branch-and-bound settings to use under the circumstances mentioned in this section.

4. Tighter formulations

When optimizer parameter settings (including aggressive application of cuts) fail to yield the desired improvements, the practitioner may obtain additional performance gains by adding cuts more specific to the model. The cuts added by the optimizer typically rely either on general polyhedral theory that applies to all MIPs, or on special structure that appears in a significant percentage of MIPs. In some cases, the cuts needed to improve performance rely on special structure specific to individual MIPs. These less applicable cuts are unlikely to be implemented in any state-of-the-art optimizer. In such cases, the practitioner may need to formulate his own cuts, drawing on specific model knowledge. One can find a staggering amount of theory on cut derivation in integer programming [20]. While more knowledge of sophisticated cut theory adds to the practitioner's quiver of tactics to improve performance, run time enhancements can be effected with some fairly simple techniques, provided the practitioner uses them in a disciplined, well organized fashion. To that end, this section describes guidelines for identifying cuts that can tighten a formulation of (P_{MIP}) and yield significant performance improvements. These guidelines can help both novice practitioners and those who possess extensive familiarity with the underlying theories of cut generation. See Rebennack et al. [21] for an example of adding cuts based on specific model characteristics.

Before tightening the formulation, the practitioner must identify elements of the model that make it difficult, specifically, those that contain the constraints and variables from which useful cuts can be derived. The following steps can help in this regard.

Determining how a MIP can be difficult to solve

- (i) *Simplify the model if necessary*. For example, try to identify any constraints or integrality restrictions that are not involved in the slow performance by systematically removing constraints and integrality restrictions and solving the resulting model. Such filtering can be done efficiently by grouping similar constraints and variables and solving model instances with one or more groups omitted. If the model remains difficult to solve after discarding a group of constraints, the practitioner can tighten the formulation without considering those constraints. Or, he can try to reproduce the problem with a smaller instance of the model.
- (ii) *Identify the constraints that prevent the objective from improving*. With a minimization problem, this typically means identifying the constraints that force activities to be performed. In other words, practical models involving nonnegative cost minimization inevitably have some constraints that prevent the trivial solution of zero from being viable.
- (iii) *Determine how removing integrality restrictions allows the root node relaxation objective to improve*. In weak formulations, the root node relaxation objective tends to be significantly better than the optimal objective of the associated MIP. The variables with fractional solutions in the root node relaxation help identify the constraints and variables that motivate additional cuts. Many models have a wealth of valid cuts that could be added purely by examining the model. But, many of those cuts may actually help little in tightening the formulation. By focusing on how relaxing integrality allows the objective to improve, the practitioner focuses on identifying the cuts that actually tighten the formulation.

Having identified the constraints and variables most likely to generate good cuts, the practitioner faces numerous ways to derive the cuts. While a sophisticated knowledge of the literature provides additional opportunities for tightening formulations, practitioners with limited knowledge of the underlying theory can still effectively tighten many formulations using some fairly simple techniques.

Model characteristics from which to derive cuts

- (i) *Linear or logical combinations of constraints* By combining constraints, one can often derive a single constraint in which fractional values can be rounded to produce a tighter cut. The clique cuts previously illustrated with the conflict graph provide an example of how to identify constraints to combine. The conflict graph in that example occurs in a sufficient number of practical MIPs so that many state-of-the-art optimizers use it. But, other MIPs may have different graphs associated with their problem structure that do not occur frequently. Identifying such graphs and implementing the associated cuts can often tighten the formulation and dramatically improve performance.
- (ii) *The optimization of one or more related models* By optimizing a related model that requires much less time to solve, the practitioner can often extract useful information to apply to the original model. For example, minimizing a linear expression involving integer variables and integer coefficients can provide a cut on that expression. This frequently helps on models with integer penalty variables.

Table 2
Under various circumstances, different formulations and algorithmic settings have a greater chance of faster solution time on an integer programming problem instance.

Characteristic	Recognition	Suggested tactic(s)
• Troublesome LPs	• Large iteration counts per node, especially regarding root node solve	• Switch algorithms between primal and dual simplex; if advanced starts do not help simplex, consider barrier method
• Lack of progress in best integer	• Little or no change in best integer solution in log after hundreds of nodes	• Use best estimate or depth-first search • Apply heuristics more frequently • Supply an initial solution • Apply discount factors in the objective • Branch up or down to resolve integer infeasibilities
• Lack of progress in best node	• Little or no change in best node in log after hundreds of nodes	• Use breadth-first search • Use aggressive probing • Use aggressive algorithmic cut generation • Apply strong branching • Derive cuts *a priori* • Reformulate with different variables
• Data and memory problems	• Slow progress in node solves • Out of memory error	• Avoid large differences in size of data • Reformulate "big *M*" constraints • Rectify LP problems, e.g., degeneracy • Apply memory emphasis setting • Buy more memory

- (iii) *Use of the incumbent solution objective value* Because cuts are often based on infeasibility, models with soft constraints that are always feasible can present unique challenges for deriving cuts. However, while any solution is feasible, the incumbent solution objective value allows the practitioner to derive cuts based on the implicit, dynamic constraint defined by the objective function and the incumbent objective value.
- (iv) *Disjunctions* Wolsey [17] provides a description of deriving cuts from disjunctions, which were first developed by Balas [15]. In general, suppose $X_1 = \{x : a^T x \geq b\}$ and $X_2 = \left\{x : \hat{a}^T x \geq \hat{b}\right\}$. Let u be the componentwise maximum of a and $\hat{a}$, i.e., $u_j = \max\{a_j, \hat{a}_j\}$. And, let $\bar{u} = \min\left\{b, \hat{b}\right\}$. Then

$$u^T x \geq \bar{u} \tag{11}$$

  is valid for $X_1 \cup X_2$, which implies it is also valid for the convex hull of X_1 and X_2. These properties of disjunctions can be used to generate cuts in practice.
- (v) *The exploitation of infeasibility* As previously mentioned, cover, clique and other cuts can be viewed as implicitly using infeasibility to identify cuts to tighten a formulation of (P_{MIP}). Generally, for any linear expression involving integer variables with integer coefficients and an integer right hand side b, if $a^T x \leq b$ can be shown to be infeasible, then the constraint $a^T x \geq b + 1$ provides a valid cut.

We now consider a simple example to illustrate the use of disjunctions to derive cuts. Most state-of-the-art optimizers support mixed integer rounding cuts, both on constraints explicitly in the model, and as Gomory cuts based on implicit constraints derived from the simplex tableau rows of the node LP subproblems. So, practitioners typically do not need to apply disjunctions to derive cuts on constraints like the one in the example we describe below. However, we use this simple example to aid in the understanding of the more challenging example we present subsequently. In the first instance, we illustrate the derivation of a mixed integer rounding cut on the constraint:

$$4x_1 + 3x_2 + 5x_3 = 10 \tag{12}$$
$$x_1, x_2, x_3 \geq 0, \quad \text{integer}. \tag{13}$$

Dividing by the coefficient of x_1, we have

$$x_1 + \frac{3}{4}x_2 + \frac{5}{4}x_3 = \frac{5}{2}. \tag{14}$$

Now, we separate the left and right hand sides into integer and fractional components, and let $\hat{x}$ represent the integer part of the left hand side:

$$\underbrace{x_1 + x_2 + x_3}_{\hat{x}} - \frac{1}{4}x_2 + \frac{1}{4}x_3 = 2 + \frac{1}{2} = 3 - \frac{1}{2}. \tag{15}$$

We examine a disjunction on the integer expression $\hat{x}$. If $\hat{x} \leq 2$, the terms with fractional coefficients on the left hand side of (15) must be greater than or equal to the first fractional term in the right-hand-side expressions. Similarly, the terms with fractional coefficients on the left hand side must be less than or equal to the second fractional term in the right-hand-side expressions if $\hat{x} \geq 3$. Using the nonnegativity of the x variables to simplify the constraints implied by the disjunction, we conclude:

$$\hat{x} \leq 2 \Rightarrow \frac{-1}{4}x_2 + \frac{1}{4}x_3 \geq \frac{1}{2} \Rightarrow x_3 \geq 2 \tag{16}$$
$$\hat{x} \geq 3 \Rightarrow \frac{-1}{4}x_2 + \frac{1}{4}x_3 \leq \frac{-1}{2} \Rightarrow x_2 \geq 2. \tag{17}$$

So, either $x_3 \geq 2$ or $x_2 \geq 2$. We can then use the result of (11) to derive the cut

$$x_2 + x_3 \geq 2. \tag{18}$$

Note that this eliminates the fractional solution $\left(2, \frac{1}{3}, \frac{1}{5}\right)$, which satisfies the original constraint, (12). Note also that by inspection the only two possible integer solutions to this constraint are $(1, 2, 0)$ and $(0, 0, 2)$. Both satisfy (18), establishing that the cut is valid. (Dividing (12) by the coefficient on x_2 or x_3 instead of x_1 results in a similar mixed integer rounding cut.)

This small example serves to illustrate the derivation of a mixed integer rounding cut on a small constraint; state-of-the-art optimizers such as CPLEX would have been able to identify this cut. However, disjunctions are more general, and can yield performance-improving cuts on models for which the optimizer's cuts do not yield sufficiently good performance. For

example, consider the following single-constraint knapsack model. Cornuejols et al. [22] originally generated this instance. (See Aardal and Lenstra [23] for additional information on these types of models.) We wish to either find a feasible solution or prove infeasibility for the single-constraint integer program:

$$13429x_1 + 26850x_2 + 26855x_3 + 40280x_4 + 40281x_5 + 53711x_6 + 53714x_7 + 67141x_8 = 45094583$$
$$x_j \geq 0, \quad \text{integer}, j = 1, \ldots, 8.$$

Running CPLEX 12.2.0.2 with default settings results in no conclusion after over 7 hours and 2 billion nodes, as illustrated in *Node Log* #7.

However, note that all the coefficients in the model are very close to integer multiples of the coefficient of x_1. Therefore, we can separate the left hand side into the part that is an integer multiple of this coefficient, and the much smaller remainder terms:

$$13429 \underbrace{(x_1 + 2x_2 + 2x_3 + 3x_4 + 3x_5 + 4x_6 + 4x_7 + 5x_8)}_{\hat{x}} \tag{19}$$
$$-8x_2 - 3x_3 - 7x_4 - 6x_5 - 5x_6 - 2x_7 - 4x_8 \tag{20}$$
$$= 3358 * 13429 + 1 = 3359 * 13429 - 13428. \tag{21}$$

This constraint resembles the one from which we previously derived the mixed integer rounding cut. But, instead of separating the integer and fractional components, we separate the components that are exact multiples of the coefficient of x_1 from the remaining terms. We now perform the disjunction on $\hat{x}$ in an analogous manner, again using the nonnegativity of the variables.

$$\hat{x} \leq 3358 \Rightarrow \underbrace{-8x_2 - 3x_3 - 7x_4 - 6x_5 - 5x_6 - 2x_7 - 4x_8}_{\leq 0} \geq 1. \tag{22}$$

Thus, if $\hat{x} \leq 3358$, the model is infeasible. Therefore, infeasibility implies that $\hat{x} \geq 3359$ is a valid cut. We can derive an additional cut from the other side of the disjunction on $\hat{x}$:

$$\hat{x} \geq 3359 \Rightarrow -8x_2 - 3x_3 - 7x_4 - 6x_5 - 5x_6 - 2x_7 - 4x_8 \leq -13428. \tag{23}$$

This analysis shows that constraints (24) (using the infeasibility argument above) and (25) (multiplying (23) through by -1) are globally valid cuts.

$$x_1 + 2x_2 + 2x_3 + 3x_4 + 3x_5 + 4x_6 + 4x_7 + 5x_8 \geq 3359 \tag{24}$$
$$8x_2 + 3x_3 + 7x_4 + 6x_5 + 5x_6 + 2x_7 + 4x_8 \geq 13428. \tag{25}$$

Adding these cuts enables CPLEX 12.2.0.2 to easily identify that the model is infeasible (see *Node Log* #8). Summarizing this example, concepts (iv) and (v), the use of disjunctions and exploiting infeasibility, helped generate cuts that turned a challenging MIP into one that was easily solved.

The second practical example we consider is a rather large maximization problem, and illustrates concepts (ii) and (v): the optimization of one or more related models and the exploitation of infeasibility, respectively. The example involves a collection of n objects with some measure of distance between them. The model selects $k < n$ of the objects in a way that maximizes the sum of the distances between the selected object, i.e., the k most diverse objects are selected. The most direct model formulation involves binary variables and a quadratic objective. Let $d_{ij} \geq 0$ be the known distance between object i and object j, and let x_i be a binary variable that is 1 if object i is selected, and 0 otherwise. The formulation follows:

$$(MIQP) \max \sum_{i=1}^{n} \sum_{j=i+1}^{n} d_{ij} x_i x_j$$
$$\text{subject to } \sum_{j=1}^{n} x_j \leq k$$
$$x_j \text{ binary}.$$

Because this article focuses on linear and linear-integer models, we consider an equivalent linear formulation that recognizes that the product of binary variables is itself a binary variable [24]. We replace each product of binaries $x_i x_j$ in (*MIQP*) with a binary variable z_{ij}, and add constraints to express the relationship between x and z in a mixed integer linear program (MILP):

$$(MILP) \max \sum_{j=1}^{n} \sum_{\substack{i=1 \\ i<j}}^{n} d_{ij} z_{ij} \tag{26}$$
$$\text{subject to } \sum_{j=1}^{n} x_j \leq k \tag{27}$$
$$z_{ij} \leq x_i \quad \forall\, i, j \tag{28}$$
$$z_{ij} \leq x_j \quad \forall\, i, j \tag{29}$$
$$x_i + x_j \leq 1 + z_{ij} \quad \forall\, i, j \tag{30}$$
$$x_j, z_{ij} \text{ binary} \quad \forall\, i, j. \tag{31}$$

The constraints (28)–(30) exist for indices (i, j), $i < j$ because the selection of both i and j is equivalent to the selection of both j and i. Hence, the model only defines z_{ij} variables with $i < j$. Note that if x_i or $x_j = 0$, then constraints (28) and (29) force z_{ij} to 0, while (30) imposes no restriction on z_{ij}. Similarly, if both x_i and $x_j = 1$, (28) and (29) impose no restriction on z_{ij}, while (30) forces z_{ij} to 1. So, regardless of the values of x_i and x_j, $z_{ij} = x_i x_j$, and we can replace occurrences of $x_i x_j$ with z_{ij} to obtain the linearized reformulation above.

This linearized model instance with $n = 60$ and $k = 24$ possesses 1830 binary variables and 5311 constraints. Due to the large branch-and-bound tree resulting from this instance, we set CPLEX's file parameter to instruct CPLEX to efficiently swap the memory associated with the branch-and-bound tree to disk. This enables the run to proceed further than with default settings in which CPLEX stores the tree in physical memory. All other parameter settings remain at defaults, so CPLEX makes use of all four available processors. CPLEX runs for just over four hours (see *Node Log* #9), terminating when the size of the swap file for the branch-and-bound tree exceeds memory limits, i.e., at the point at which CPLEX has processed over 4 million nodes and the solution has an objective value of 3483.0000, proven to be within 51.32% of optimal. This level of performance indicates significant potential for improvement. Although we do not provide the output here, the original MIQP formulation in (*MIQP*) performs even worse.

Experimentation with non-default parameter settings as described in Section 3 yields modest performance improvements, but does not come close to enabling CPLEX to find an optimal solution to the model.

We carefully examine a smaller model instance with $n = 3$ and $k = 2$ to assess how removing integrality restrictions yields an artificially high objective function value:

Node Log #7

```
           Nodes                                         Cuts/
      Node    Left     Objective  IInf  Best Integer     Best Node          ItCnt     Gap
...
 2054970910 13066        0.0000     1                      0.0000       25234328
Elapsed real time = 27702.98 sec. (tree size =  2.70 MB, solutions = 0)
 2067491472 14446        0.0000     1                      0.0000       25388082
 2080023238 12892        0.0000     1                      0.0000       25542160
 2092548561 15366        0.0000     1                      0.0000       25696280
...
                          -------
Total (root+branch&cut) = 28302.29 sec.


MIP - Node limit exceeded, no integer solution.
Current MIP best bound =  0.0000000000e+00 (gap is infinite)
Solution time = 28302.31 sec.  Iterations = 25787898  Nodes = 2100000004 (16642)
```

Node Log #8

```
        Nodes                                          Cuts/
   Node  Left     Objective  IInf  Best Integer     Best Node    ItCnt     Gap

      0     0        0.0000     1                      0.0000        1
      0     0        0.0000     2                  MIRcuts: 1        3
      0     0        0.0000     2                  MIRcuts: 1        5
      0     0        cutoff                                       5
Elapsed real time =   0.23 sec. (tree size =  0.00 MB, solutions = 0)
Mixed integer rounding cuts applied:  1
...
MIP - Integer infeasible.
Current MIP best bound is infinite.
Solution time =    0.46 sec.  Iterations = 5  Nodes = 0
```

Node Log #9

```
        Nodes                                          Cuts/
   Node  Left     Objective  IInf  Best Integer     Best Node    ItCnt     Gap

*     0+    0                            0.0000                 2247      ---
      0     0     7640.4000  1830         0.0000     7640.4000     2247      ---
*     0+    0                           19.0000     7640.4000     2247      ---

...

*     0+    0                         3185.0000     7445.4286     2286  133.77%
      0     2     7628.5333  1829     3185.0000     7445.4286     2286  133.77%
Elapsed real time =   4.09 sec. (tree size =  0.01 MB, solutions = 8)
     35    37     6579.2308  1378     3185.0000     7445.4286     6615  133.77%
...
 4332613 3675298     4936.6750  1099     3483.0000     5270.8377 1.78e+08 51.33%
 4341075 3682375     3889.4643   714     3483.0000     5270.4545 1.79e+08 51.32%

...
CPLEX Error  1803: Failure on temporary file write.

Solution pool: 25 solutions saved.

MIP - Error termination, no tree:  Objective =  3.4830000000e+03
Current MIP best bound =  5.2704102564e+03 (gap = 1787.41, 51.32%)
Solution time = 15031.18 sec.  Iterations = 178699476  Nodes = 4342299 (3682262)
```

$$\max 3z_{12} + 4z_{13} + 5z_{23}$$
$$\text{subject to } x_1 + x_2 + x_3 \leq 2$$
$$z_{12} - x_1 \leq 0$$
$$z_{12} - x_2 \leq 0$$
$$x_1 + x_2 \leq 1 + z_{12}$$
$$z_{13} - x_1 \leq 0$$
$$z_{13} - x_3 \leq 0$$
$$x_1 + x_3 \leq 1 + z_{13}$$
$$z_{23} - x_2 \leq 0$$
$$z_{23} - x_3 \leq 0$$
$$x_2 + x_3 \leq 1 + z_{23}$$
$$x_1, x_2, x_3, z_{12}, z_{13}, z_{23} \text{ binary.}$$

The optimal solution to this MILP consists of setting $z_{23} = x_2 = x_3 = 1$, yielding an objective value of 5. By contrast, relaxing integrality enables a fractional solution consisting of setting all x and z variables to 2/3, yielding a much better objective value of 8. Note that the difference between the MILP and its relaxation occurs when the z_{ij} variables assume values strictly less than 1. When any $z_{ij} = 1$, the corresponding x_i and x_j variables are forced to 1 by constraints (28) and (29) for both the MILP and its LP relaxation. By contrast, when $0 \leq z_{ij} < 1$, x_i or x_j must assume a value of 0 in the MILP, but not in the relaxation. Thus, in the LP relaxation, we can set more of the z variables to positive values than in the MILP. This raises the question of how many z variables we can set to 1 in the MILP. In the optimal solution, only z_{23} assumes a value of 1. So, can we set two of the z variables to 1 and find a feasible solution to the MILP? To answer this question, pick any two z variables and set them to 1. Since each z variable is involved in similar types of constraints, without loss of generality, we set z_{12} and z_{13} to 1. From the constraints:

$$z_{12} - x_1 \leq 0$$
$$z_{12} - x_2 \leq 0$$
$$z_{13} - x_1 \leq 0$$
$$z_{13} - x_3 \leq 0$$

we see that x_1, x_2, and x_3 must all be set to 1. But this violates the constraint that the x variables can sum to at most 2. For any of the other two distinct pairs of z variables in this smaller model, all three x variables are forced to a value of 1 since for the MILP:

$$z_{ij} > 0 \Longleftrightarrow x_i = x_j = 1. \tag{32}$$

Thus, any distinct pair of z variables set to 1 forces three x variables to 1, violating the constraint that $x_1 + x_2 + x_3 \leq 2$. Hence, in any integer feasible solution, at most one z variable can be set to 1. This implies that the constraint:

$$z_{12} + z_{13} + z_{23} \leq 1$$

is a globally valid cut. And, we can see that it cuts off the optimal solution of the LP relaxation consisting of setting each z variable to 2/3.

We now generalize this to (*MILP*), in which the x variables can sum to at most k. We wish to determine the number of z variables we can set to 1 in (*MILP*) without forcing the sum of the x variables to exceed k. Suppose we set k of the x variables to 1. Since (32) holds for all pairs of x variables, without loss of generality, consider an integer feasible solution with $x_1 = x_2 = \cdots = x_k = 1$, and $x_{k+1} = \cdots = x_n = 0$. From (32), $z_{ij} = 1$ if and only if $1 \leq i \leq k$, $1 \leq j \leq k$, and $i < j$. We can therefore count the number of z variables that equal 1 when $x_1 = x_2 = \cdots = x_k = 1$. Specifically, there are $k(k-1)$ pairs (i, j) with $i \neq j$, but only half of them have $i < j$. So, at most $k(k-1)/2$ of the z_{ij} variables can be set to 1 when k of the x variables are set to 1. In other words,

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} z_{ij} \leq k(k-1)/2$$

is a globally valid cut.

Adding this cut to the instance with $n = 60$ and $k = 24$ enables CPLEX to solve the model to optimality in just over 2 hours and 30 min on the same machine using settings identical to those from the previous run without the cut. (See *Node Log* #10.) Note that the cut tightened the formulation significantly, as can be seen by the much better root node objective value of 4552.4000, which compares favorably to the root node objective value of 7640.4000 on the instance without the cut. Furthermore, the cut enabled CPLEX to add numerous zero-half cuts to the model that it could not with the original formulation. The zero-half cuts resulted in additional progress in the best node value that was essential to solving the model to optimality in a reasonable amount of time.

Given the modest size of the model, a run time of 2.5 hours to optimality suggests potential for additional improvements in the formulation. However, by adding one globally valid cut, we see a dramatic performance improvement nonetheless. Furthermore, the derivation of this cut draws heavily on the guidelines proposed for tightening the formulation. By using a small instance of the model, we can easily identify how removal of integrality restrictions enables the objective to improve. Furthermore, we use infeasibility to derive the cut: by recognizing that the simplified MILP model is infeasible when $z_{12} + z_{13} + z_{23} \geq 2$, we show that $z_{12} + z_{13} + z_{23} \leq 1$ is a valid cut.

5. Conclusion

Today's hardware and software allow practitioners to formulate and solve increasingly large and detailed models. However, optimizers have become less straightforward, often providing many methods for implementing their algorithms to enhance performance given various mathematical structures. Additionally, the literature regarding methods to increase the tractability of mixed integer linear programming problems contains a high degree of theoretical sophistication. Both of these facts might lead a practitioner to conclude that developing the skills necessary to successfully solve difficult mixed integer programs is too time consuming or difficult. This paper attempts to refute that perception, illustrating that practitioners can implement many techniques for improving performance without expert knowledge in the underlying theory of integer programming, thereby enabling them to solve larger and more detailed models with existing technology.

Acknowledgments

Dr. Klotz wishes to acknowledge all of the CPLEX practitioners over the years, many of whom have provided the wide variety of models that revealed the guidelines described in this paper. He also wishes to thank the past and present CPLEX development, support, and sales and marketing teams who have contributed to the evolution of the product. Professor Newman wishes to thank former doctoral students Chris Cullenbine, Brian Lambert, Kris Pruitt, and Jennifer Van Dinter at the Colorado School of Mines for their helpful comments; she also wishes to thank her colleagues Jennifer Rausch (Jeppeson, Englewood, Colorado) and Professor Josef Kallrath (BASF-AG, Ludwigshafen, Germany) for helpful comments on an earlier draft. Both authors thank an anonymous referee for his helpful comments that improved the paper.

Node Log #10

```
        Nodes                                         Cuts/
   Node  Left     Objective  IInf  Best Integer    Best Node    ItCnt     Gap

*     0+    0                           0.0000                  1161      ---
      0     0     4552.4000   750        0.0000     4552.4000     1161      ---
*     0+    0                           6.0000     4552.4000     1161      ---
...
*     0+    0                        3477.0000     3924.7459    37882   12.88%
      0     2     3924.7459  1281     3477.0000     3924.7459    37882   12.88%
Elapsed real time =  51.42 sec. (tree size =  0.01 MB, solutions = 31)
      1     3     3919.3378  1212     3477.0000     3924.7459    39886   12.88%
      2     4     3910.8201  1243     3477.0000     3924.7459    42289   12.88%
      3     5     3910.8041  1144     3477.0000     3919.3355    44070   12.72%
...
 125571  7819        cutoff           3590.0000     3599.7046 60456851    0.27%
Elapsed real time = 9149.19 sec. (tree size = 234.98 MB, solutions = 43)
Nodefile size = 196.38 MB (168.88 MB after compression)
*126172  7231      integral     0     3591.0000     3599.7046 60571398    0.24%
 127700  5225        cutoff           3591.0000     3598.0159 60769494    0.20%
 131688     6        cutoff           3591.0000     3592.5939 60980430    0.04%

Zero-half cuts applied:  2244

Solution pool: 44 solutions saved.

MIP - Integer optimal solution:  Objective =  3.5910000000e+03
Solution time = 9213.79 sec.  Iterations = 60980442  Nodes = 131695
```

Both authors wish to remember Lloyd Clarke (February 14, 1964–September 20, 2007). His departure from the CPLEX team had consequences that extended beyond the loss of an important employee and colleague.

References

[1] R. Bixby, E. Rothberg, Progress in computational mixed integer programming — a look back from the other side of the tipping point, Annals of Operations Research 149 (1) (2007) 37–41.
[2] IBM, 2012. ILOG CPLEX. Incline Village, NV.
[3] Gurobi, 2012. Gurobi Optimizer. Houston, TX.
[4] MOPS, 2012. MOPS. Paderborn, Germany.
[5] MOSEK, 2012. MOSEK Optimization Software. Copenhagen, Denmark.
[6] FICO, 2012. Xpress-MP Optimization Suite. Minneapolis, MN.
[7] E. Klotz, A. Newman, Practical guidelines for solving difficult linear programs, Surveys in Operations Research and Management Science 18 (1–2) (2013) 1–17.
[8] R. Rardin, Optimization in Operations Research, Prentice Hall, 1998, Ch. 6.
[9] W. Winston, Operations Research: Applications and Algorithms, Brooks/Cole, Thompson Learning, 2004.
[10] M. Savelsbergh, Preprocessing and probing techniques for mixed integer programming problems, INFORMS Journal on Computing 6 (4) (1994) 445–454.
[11] E. Balas, An additive algorithm for solving linear programs with zero-one variables, Operations Research 13 (4) (1965) 517–546.
[12] E. Danna, Performance variability in mixed integer programming. Presentation at MIP 2008 Workshop, Columbia University, 2008.
[13] T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R. Bixby, E. Danna, G. Gamrath, A. Gleixner, S. Heinz, A. Lodi, H. Mittelmann, T. Ralphs, D. Salvagnin, D. Steffy, K. Wolter, MIPLIB 2010, Mathematical Programming Computation 3 (2) (2011) 103–163. URL http://mpc.zib.de/index.php/MPC/article/view/56/28.
[14] W. Lambert, A. Brickey, A. Newman, K. Eurek, Open pit block sequencing formulations: a tutorial, Interfaces (in press).
[15] E. Balas, Disjunctive programming: properties of the convex hull of feasible points, Discrete Applied Mathematics 89 (1998) 3–44.
[16] T. Achterberg, Constraint Integer Programming (Ph.D. Dissertation). Technical University Berlin, Berlin, 2007.
[17] L. Wolsey, Integer Programming, Wiley, 1998.
[18] D. Bertsimas, S. Stock Patterson, The air traffic flow management problem with enroute capacities, Operations Research 46 (3) (1998) 406–422.
[19] J. Camm, A. Raturi, S. Tadisina, Cutting big M down to size, Interfaces 20 (5) (1990) 61–66.
[20] Grötschel (Ed.), The Sharpest Cut: The Impact of Manfred Padberg and His Work, in: MPS-SIAM Series on Optimization, 2004.
[21] S. Rebennack, G. Reinelt, P. Pardalos, A tutorial on branch and cut algorithms for the maximum stable set problem, International Transactions in Operational Research 19 (1–2) (2012) 161–199. URL http://dx.doi.org/10.1111/j.1475-3995.2011.00805.x.
[22] G. Cornuejols, R. Urbaniak, R. Weismantel, L. Wolsey, Decomposition of integer programs and of generating sets, in: Proceedings of the 5th Annual European Symposium on Algorithms 1284, in: Lecture Notes in Computer Science, 1997, pp. 92–102.
[23] K. Aardal, A. Lenstra, Hard equality constrained integer knapsacks, Mathematics of Operations Research 3 (29) (2004) 724–738.
[24] L. Watters, Reduction of integer polynomial programming to zero-one linear programming problems, Operations Research 15 (6) (1967) 1171–1174.

Surveys in Operations Research and Management Science 18 (2013) 33–53

Contents lists available at ScienceDirect

Surveys in Operations Research and Management Science

journal homepage: www.elsevier.com/locate/sorms

Review

The fundamental risk quadrangle in risk management, optimization and statistical estimation☆

R. Tyrrell Rockafellar[a], Stan Uryasev[b,*]

[a] University of Washington, Department of Mathematics, Box 354350, Seattle, WA 98195-4350, United States
[b] University of Florida, ISE Department, P.O. Box 116595, 303 Weil Hall, Gainesville, FL 32611-6595, United States

ARTICLE INFO

Article history:
Received 3 March 2013
Accepted 3 March 2013

ABSTRACT

Random variables that stand for cost, loss or damage must be confronted in numerous situations. Dealing with them systematically for purposes in risk management, optimization and statistics is the theme of this presentation, which brings together ideas coming from many different areas.

Measures of risk can be used to quantify the hazard in a random variable by a single value which can substitute for the otherwise uncertain outcomes in a formulation of constraints and objectives. Such quantifications of risk can be portrayed on a higher level as generated from penalty-type expressions of "regret" about the mix of potential outcomes. A trade-off between an up-front level of hazard and the uncertain residual hazard underlies that derivation. Regret is the mirror image of utility, a familiar concept for dealing with gains instead of losses, but regret concerns hazards relative to a benchmark. It bridges between risk measures and expected utility, thereby reconciling those two approaches to optimization under uncertainty.

Statistical estimation is inevitably a partner with risk management in handling hazards, which may be known only partially through a data base. However, a much deeper connection has come to light with statistical theory itself, in particular regression. Very general measures of error can associate with any hazard variable a "statistic" along with a "deviation" which quantifies the variable's nonconstancy. Measures of deviation, on the other hand, are paired closely with measures of risk exhibiting "aversity". A direct correspondence can furthermore be identified between measures of error and measures of regret. The fundamental quadrangle of risk developed here puts all of this together in a unified scheme.

Contents

1. Introduction

The challenges of dealing with risk pervade many areas of management and engineering. The decisions that have to be made in risky situations must nonetheless confront constraints on their consequences, no matter how uncertain those consequences may be. Furthermore, the decisions need to be open to comparisons which enable some kind of optimization to take place.

When uncertainty is modeled probabilistically with random variables, practical challenges arise about estimating properties of those random variables and their interrelationships. Information may come from empirical distributions generated by sampling, or there may only be databases representing information accumulated somehow or other in the past. Standard approaches to statistical analysis and regression in terms of expectation, variance and covariance may then be brought in. But the prospect is now emerging of a vastly expanded array of tools which can be finely tuned to

☆ This research has been supported by the AFOSR grant FA9550-11-1-0258, "New Developments in Uncertainty: Linking Risk Management, Reliability, Statistics and Stochastic Optimization.

* Corresponding author. Tel.: +1 352 213 3457.
E-mail addresses: rtr@uw.edu (R.T. Rockafellar), uryasev@ufl.edu (S. Uryasev).

1876-7354/$ – see front matter © 2013 Published by Elsevier B.V.
http://dx.doi.org/10.1016/j.sorms.2013.03.001

reflect the various ways that risk may be assessed and, at least to some extent, controlled.

$$\begin{array}{ccccc} & & \text{risk } \mathcal{R} & \longleftrightarrow & \mathcal{D} \text{ deviation} \\ \text{optimization} & & \uparrow\downarrow \ \mathcal{S} & & \downarrow\uparrow & & \text{estimation} \\ & & \text{regret } \mathcal{V} & \longleftrightarrow & \mathcal{E} \text{ error} \end{array}$$

Diagram 1: The Fundamental Risk Quadrangle.

This paper is aimed at promoting and developing such tools in a new paradigm we call the *risk quadrangle*, which is shown in Diagram 1. It brings together several lines of research and methodology which, until now, been pursued separately in different professional areas with little inkling of their fertile interplay. The ideas in these areas form such a vast subject that a broad survey with full references is beyond feasibility. Our contribution here must, in part, be seen therefore as providing an overview of the connections, supplemented by instructive examples and the identification of issues in need of more attention. However, many new facts are brought to light along with new results and broad extensions of earlier results.

$\mathcal{R}(X)$ provides a numerical surrogate for the overall hazard in X,
$\mathcal{D}(X)$ measures the "nonconstancy" in X as its uncertainty,
$\mathcal{E}(X)$ measures the "nonzeroness" in X,
$\mathcal{V}(X)$ measures the "regret" in facing the mix of outcomes of X,
$\mathcal{S}(X)$ is the "statistic" associated with X through $\mathcal{E}$ and $\mathcal{V}$.

Diagram 2: The Quantifications in the Quadrangle.

The context is that of random variables that can be thought of as standing for uncertain "costs" or "losses" in the broadest sense, not necessarily monetary (with a negative "cost" corresponding perhaps to a "reward"). The language of cost gives the orientation that we would like the outcomes of these random variables to be lower rather than higher, or to be held below some threshold. All sorts of indicators that may provide signals about hazards can be viewed from this perspective. The quadrangle elements provide numerical "quantifications" of them (not only finite numbers but in some cases ∞) which can be employed for various purposes.

It will help, in understanding the quadrangle, to begin at the upper left corner, where $\mathcal{R}$ is a so-called *measure of risk*. The specific sense of this needs clarification, since there are conflicting angles to the meaning of "risk". In denoting a random cost by X and a constant by C, a key question is how to give meaning to a statement that X is "adequately" $\leq C$ with respect to the preferences of a decision maker who realizes that uncertainty might inescapably generate some outcomes of X that are $>C$. The role of a risk measure $\mathcal{R}$, in the sense intended here, is to answer this question by aggregating the overall uncertain cost in X into a single numerical value $\mathcal{R}(X)$ in order to

model "X adequately $\leq C$" by the inequality $\mathcal{R}(X) \leq C$.

There are familiar ways of doing this. One version could be that X is $\leq C$ on average, as symbolized by $\mu(X) \leq C$ with $\mu(X)$ the mean value, or in equivalent notation (both are convenient to maintain), $EX \leq C$ with EX the expected value. Then $\mathcal{R}(X) = \mu(X) = EX$. A tighter version could be $\mu(X) + \lambda\sigma(X) \leq C$ with λ giving a positive multiple of the standard deviation $\sigma(X)$ so as to provide a safety margin reminiscent of a confidence level in statistics; then $\mathcal{R}(X) = \mu(X)+\lambda\sigma(X)$. The alternative idea that the inequality should hold at least with a certain probability $\alpha \in (0, 1)$ corresponds to $q_\alpha(X) \leq C$ with $q_\alpha(X)$ denoting the α-quantile of X, whereas insisting that $X \leq C$ almost surely can be written as $\sup X \leq C$ with $\sup X$ standing for the essential supremum of X. Then $\mathcal{R}(X) = q_\alpha(X)$ or $\mathcal{R}(X) = \sup X$, respectively.[1] However, these examples are just initial possibilities among many for which pros and cons need to be appreciated.

A typical situation in optimization that illustrates the compelling need for measures of risk revolves around a family of random "costs" that depend on a decision vector x belonging to a subset $S \subset \mathbb{R}^n$,

$$X_i(x) \quad \text{for } i = 0, 1, \ldots, m, \text{ where } x = (x_1, \ldots, x_n). \tag{1.1}$$

The handicap is that x can usually do no more than influence the probability distribution of each of the "costs". A potential aim in choosing x from S would be to keep the random variable $X_i(x)$ adequately $\leq c_i$ for $i = 1, \ldots, m$, while achieving the lowest c_0 such that $X_0(x)$ is adequately $\leq c_0$. The way "adequately" is modeled could be different for each i, and the notion of a risk measure provides the perfect tool. A selection of risk measures $\mathcal{R}_i$ that pins down the intended sense of "adequately" in each case leads a optimization problem having the form

$$\begin{aligned} &\text{choose } x \in S \text{ to minimize } \mathcal{R}_0(X_0(x)) \\ &\quad \text{subject to } \mathcal{R}_i(X_i(x)) \leq c_i \quad \text{for } i = 1, \ldots, m. \end{aligned} \tag{1.2}$$

Besides pointing the way toward risk-oriented problem formulations to which optimization technology can successfully be applied, this illustration brings another issue to the foreground. In selecting a measure of risk $\mathcal{R}_i$, it may not be enough just to rely on $\mathcal{R}_i$ having an appealing interpretation. An important consideration may be whether $\mathcal{R}_i$ produces expressions $\mathcal{R}_i(X_i(x))$ that behave reasonably as functions of $x = (x_1, \ldots, x_n)$. Axioms laying out sensible standards for a measure of risk, such the *coherency* introduced in [1], are vital for that.[2]

Another idea in dealing with uncertainty in a random variable X is to quantify its nonconstancy through a *measure of deviation* $\mathcal{D}$, with $\mathcal{D}(X)$ then being a generalization of $\sigma(X)$. Again, axioms have to be articulated. The distinction between $\mathcal{D}$ and $\mathcal{R}$ at the top of the quadrangle is essential, despite a very close connection, because of differences in axioms and roles played in applications.

Motivation for nonstandard measures of deviation is apparent in particular in finance because of the heavy concentration there on variance – or equivalently standard deviation – despite shortcomings in capturing dangerous "tail behavior" in probability distributions. In portfolio theory, the rate of return of the portfolio is a random variable $X(x)$ depending on the vector x that gives the proportions of various securities included in the portfolio. Bounds are placed on $\sigma(X(x))$ or this quantity is minimized subject to side conditions on x. Such an approach can be justified when the random variables have normal distributions, but when the heavy tail behavior of nonnormal distributions enters the scene, doubts arise. It may be better then to replace standard deviation by a different deviation measure, which perhaps could even act on $X(x)$ asymmetrically.[3]

The introduction of nonstandard deviation measures $\mathcal{D}$ in place of σ brings up the question of whether this might entail some kind of generalization in statistical theory itself. That is indeed one of the

[1] Note that $\mathcal{R}(X) = \sup X$ gives examples where $\mathcal{R}(X)$ might be ∞.

[2] The axioms will be developed in Section 3 and their consequences for optimization problems like (1.2) fully pinned down in Section 5.

[3] The "two fund theorem" and other celebrated results of portfolio theory that revolve around standard deviation can be extended in this direction with CAPM equations replaced by other equations derived from alternative measures of deviation; cf. [2,3].

questions our quadrangle scheme is aimed at answering, as will be explained shortly.[4]

We turn now to the lower left corner of the quadrangle. In a *measure of regret* $\mathcal{V}$, the value $\mathcal{V}(X)$ stands for the net displeasure perceived in the potential mix of outcomes of a random "cost" X which may sometimes be >0 (bad) and sometimes ≤ 0 (OK or better).[5,6] Regret comes up in penalty approaches to constraints in stochastic optimization and, in mirror image, corresponds to measures of "utility" $\mathcal{U}$ in a context of gains Y instead of losses X, which is typical in economics: $\mathcal{V}(X) = -\mathcal{U}(-X)$, $\mathcal{U}(Y) = -\mathcal{V}(-Y)$. Regret obeys $\mathcal{V}(0) = 0$, so in this pairing we have to focus on utility measures that have $\mathcal{U}(0) = 0$; we say then that $\mathcal{U}$ is a *measure of relative utility*. The interpretation is that, in applying $\mathcal{U}$ to Y, we are thinking of Y not as absolute gain but gain relative to some threshold, e.g., $Y = Y_0 - B$ where Y_0 is absolute gain and B is a benchmark.

Focusing on relative utility in this sense is a positive feature of the quadrangle scheme because it can help to capture the sharp difference in attitude toward outcomes above or below a benchmark that is increasingly acknowledged as influencing the preferences of decision makers.[7]

Measures of regret $\mathcal{V}$, like measures of deviation $\mathcal{D}$, are profoundly related to measures of risk $\mathcal{R}$, and one of our tasks will to bring this all out. Especially important will be a one-to-one correspondence between measures of deviation and measures of risk under "aversity", regardless of coherency. A powerful property of measures of regret, which soon will be discussed, is their ability to generate measures of risk through trade-off formulas. By means of such formulas, an optimization problem in the form of (1.2) may be recast in terms of regret instead of risk, and this can be a great simplification.[8]

Furthermore, by revealing a deep connection between risk measures and utility, regret reconciles the seemingly different approaches to optimization based on those concepts.

The interesting question already raised, of whether measures of deviation beyond standard deviation might fit into some larger development in statistical theory, is our next topic. It brings us to the lower right corner of the quadrangle, where we speak of a *measure of error* $\mathcal{E}$ as assigning to a random variable X a value $\mathcal{E}(X)$ that quantifies the nonzeroness in X. Classical examples are the $\mathcal{L}^p$-norms

$$\|X\|_1 = E|X|, \qquad \|X\|_p = [E(|X|^p)]^{1/p} \quad \text{for } p \in (1,\infty),$$
$$\|X\|_\infty = \sup|X|, \tag{1.3}$$

but there is much more to think of besides norms.

Given an error measure $\mathcal{E}$ and a random variable X, one can look for a constant C nearest to X in the sense of minimizing $\mathcal{E}(X-C)$. The resulting minimum "$\mathcal{E}$-distance", denoted by $\mathcal{D}(X)$, turns out to be a deviation measure (under assumptions explained later). The C value in the minimum, denoted by $\mathcal{S}(X)$, can be called the "*statistic*" associated with X by $\mathcal{E}$. The case of $\mathcal{E}(X) = \|X\|_2$ produces $\mathcal{S}(X) = EX$ and $\mathcal{D}(X) = \sigma(X)$, but many other examples will soon be seen.

The emergence of a particular deviation measure $\mathcal{D}$ and statistic $\mathcal{S}$ from the choice of an error measure $\mathcal{E}$ has intriguing implications for statistical estimation in the sense of generalized regression. There is furthermore a deep connection between regression and an optimization problem like (1.2). The x-dependent random variables $X_i(x)$ there might be replaced by convenient approximations $\hat{X}_i(x)$ developed through regression, and the particular mode of regression might have significant consequences. We will get back to this shortly.

The optimization and estimation sides of the quadrangle are bound together not only through such considerations, but also in a more direct manner. The rule that projects from $\mathcal{E}$ onto $\mathcal{D}$ is echoed by a certainty–uncertainty *trade-off formula* which projects a regret measure $\mathcal{V}$ onto a risk measure $\mathcal{R}$. This formula, in which $C + \mathcal{V}(X - C)$ is minimized over C, generalizes a rule in [5,6], for VaR–CVaR computations. It extends the insights gained beyond that by Ben-Tal and Teboulle [7] in a context of expected utility, and lines up with still broader expressions for risk in [8]. Under a simple relationship between $\mathcal{V}$ and $\mathcal{E}$, the optimal C value in the trade-off is the same statistic $\mathcal{S}(X)$ as earlier, but that conceptual bond has been missed. Nothing has hitherto suggested that "error" in its context of approximation might be inherently related to the very different concept of "regret" and, through that, to "utility".

Altogether, we arrive in this way at a "quadrangle" of quantifications having the descriptions in Diagram 2 and the interconnections in Diagram 3.[9] More details will be furnished in Section 3, after the assumptions needed to justify the relationships have been explained.

$$\mathcal{R}(X) = EX + \mathcal{D}(X), \qquad \mathcal{D}(X) = \mathcal{R}(X) - EX$$
$$\mathcal{V}(X) = EX + \mathcal{E}(X), \qquad \mathcal{E}(X) = \mathcal{V}(X) - EX$$
$$\mathcal{R}(X) = \min_C \{C + \mathcal{V}(X-C)\}, \qquad \mathcal{D}(X) = \min_C \{\mathcal{E}(X-C)\}$$
$$\operatorname*{argmin}_C \{C + \mathcal{V}(X-C)\} = \mathcal{S}(X) = \operatorname*{argmin}_C \{\mathcal{E}(X-C)\}$$

Diagram 3: The General Relationships.

The paired arrows on the sides of Diagram 1, in contrast to the two-way arrows on the top and bottom, correspond to the fact that the simple formulas in Diagram 3 for getting $\mathcal{R}$ and $\mathcal{D}$ from $\mathcal{V}$ and $\mathcal{E}$ are not uniquely invertible. Antecedents $\mathcal{V}$ and $\mathcal{E}$ for $\mathcal{R}$ and $\mathcal{D}$ always exist, even in multiplicity,[10] so the real issue for inversion is the identification of *natural, nontrivial* antecedents. That is a large topic with many good answers in the examples in Section 2 and broader principles in Section 3.

More must be said now about how the quadrangle relates to statistical estimation in the form of regression, and the motivation coming from that. Broader approaches to regression than classical "least-squares" are not new, but the description to be given here is unprecedently broad.

Regression is a way of approximating a random variable Y by a function $f(X_1, \ldots, X_n)$ of one or more other random variables X_j for purposes of anticipating outcome properties or trends. It requires a way of measuring how far the random difference $Z_f = Y - f(X_1, \ldots, X_n)$ is from 0. That, clearly, is where error measures $\mathcal{E}$ can come in. The norms (1.3) offer choices, but there may be incentive for using asymmetric error measures $\mathcal{E}$ that look at more than just $|Z_f|$. When Y has cost or hazard orientation, underestimations $Y -$

[4] Nonstandard deviation measures are also connected to statistics through entropy analysis, cf. [4].

[5] Regret in this sense is distinct from the notion of regret as "opportunity loss" in some versions of decision theory.

[6] In financial terms, if X and $\mathcal{V}(X)$ have units of money, $\mathcal{V}(X)$ can be the compensation deemed appropriate for taking on the burden of the uncertain loss X.

[7] This will be discussed in more detail in Section 4 in the case of utility expressions $\mathcal{U}(Y) = E[u(Y)]$ for an underlying function u. Having $\mathcal{U}(0) = 0$ corresponds to having $u(0) = 0$, which can be achieved by selecting a benchmark and shifting the graph of a given "absolute" utility so that benchmark point is at the origin of $\mathbb{R}^2$.

[8] This is mission of the Regret Theorem in Section 5.

[9] The "argmin" notation refers to the C values that achieve the "min".

[10] For instance $\mathcal{V}(X) = \mathcal{R}(X) + \alpha|EX|$ and $\mathcal{E}(X) = \mathcal{D}(X) + \alpha|EX|$ with $\alpha > 0$.

$f(X_1, \ldots, X_n) > 0$ may be more dangerous than overestimations $Y - f(X_1, \ldots, X_n) < 0$.

For an error measure $\mathcal{E}$ and a collection $\mathcal{C}$ of regression functions f, the basic problem of regression for Y with respect to $X_1, \ldots, X_n$ is to

$$\text{minimize } \mathcal{E}(Z_f) \text{ over } f \in \mathcal{C}, \quad \text{where } Z_f = Y - f(X_1, \ldots, X_n). \tag{1.4}$$

An immediate question that comes to mind is how one such version of regression might differ from another and perhaps be better for some purpose. We provide a simple but revealing answer. As long as $\mathcal{C}$ has the property that it includes with each f all the translates $f + C$ for constants C, problem (1.4) has the following interpretation:

$$\text{minimize } \mathcal{D}(Z_f) \text{ over all } f \in \mathcal{C} \quad \text{such that } \mathcal{S}(Z_f) = 0, \tag{1.5}$$

where $\mathcal{D}$ and $\mathcal{S}$ are the deviation measure and statistic associated with the error measure $\mathcal{E}$. In such generality, and with additional features as well,[11] this is a new result, but it builds in part on our earlier theorem in [9] for the case of linear regression functions f.

Factor models for simplifying work with random variables ordinarily rely on standard least-squares regression, which corresponds here to $\mathcal{E}$ being the $\mathcal{L}^2$ norm in (1.3), so that $\mathcal{D}(Z_f)$ is $\sigma(Z_f)$ and $\mathcal{S}(Z_f)$ is $\mu(Z_f)$. Suppose, for instance, that the "costs" in (1.1) have the form

$$X_i(x) = g_i(x, V_1, \ldots, V_r) \quad \text{with respect to random variables } V_k. \tag{1.6}$$

The random variables V_k may have various interdependences which can be treated by thinking of them as reflecting the outcomes of certain other, more "primitive", random variables $W_1, \ldots, W_s$. This can suggest approximating them through regression as

$$V_k \approx \hat{V}_k = f_k(W_1, \ldots, W_s) \quad \text{for } f_k \in \mathcal{C}_k \text{ and an error measure } \mathcal{E}_k, \tag{1.7}$$

which leads to approximating $X_i(x)$ by

$$\hat{X}_i(x) = g_i(x, \hat{V}_1, \ldots, \hat{V}_r) \quad \text{for } i = 0, 1, \ldots, m. \tag{1.8}$$

In the optimization problem (1.2), this replaces the objective and constraint functions $\mathcal{R}_i(X_i(x))$ by different functions $\mathcal{R}_i(\hat{X}_i(x))$. How will that change the solution? What guarantee is there that a solution to the altered problem will be close to a solution to the original problem?

That question has received very little attention so far, although we raised it in [9] as suggesting that the error measures $\mathcal{E}_k$ in (1.7) should be "tuned" somehow to the quantification of risk by $\mathcal{R}_i$. We did show there, at least, that if $g_i(x, V_1, \ldots, V_m) = x_1 V_1 + \cdots + x_m V_m$, the $\mathcal{E}_k$'s should be the error measure $\mathcal{E}_i$ in the same quadrangle as the risk measure $\mathcal{R}_i$. Then the expressions $\mathcal{R}_i(X_i(x))$ and $\mathcal{R}_i(\hat{X}_i(x))$ will be closer to each other as functions of x than otherwise. Although we do not pursue that further in this survey, we hope that the quadrangle framework we furnish will help to stimulate more research on the matter.

In the plan of the paper after this introduction, we will first pass in Section 2 to examples of quadrangles that help to underscore our intentions and provide guidance for theory and applications. This is a compromise in which we, and the reader, are held back to some extent by the postponement of precise definitions and assumptions that only come in Section 3. It is an unusual way of proceeding, but we take this path from the conviction that providing motivation in advance of technical details is essential for conveying the attractions of this wide-ranging subject.

Section 3 showcases the Quadrangle Theorem which supports the formulas and relationships in Diagrams 1–3 and specifies the key properties of the quantifiers $\mathcal{R}$, $\mathcal{D}$, $\mathcal{V}$ and $\mathcal{E}$ that propagate through the scheme. Although some connections have already been indicated elsewhere, this result is new in its generality and creation of the entire quadrangle with $\mathcal{V}$ and its associated utility $\mathcal{U}$. Also new in Section 3 in similar degree are the Scaling Theorem, the Mixing Theorem and the Reverting Theorem, which furnish means of constructing additional instances of quadrangles from known ones.

Interpretations and results beyond the basics in Section 3 are provided in Section 4 as an aid to more specialized applications. The main contribution there is the Expectation Theorem, concerned with the "expectation quadrangles" we are about to describe. In particular, it enables us to justify a number of the examples in Section 2 and show how they can be extended. Section 5 presents in more detail the role of the risk quadrangle in applications to optimization, as in problem (1.2), and generalized regression as in problem (1.4).[12] The Convexity Theorem indicates how "convex dependence" of the random variables $X_i(x)$ in (1.1) with respect to x passes over to convexity of the expressions $\mathcal{R}_i(X_i(x))$ in (1.2) under natural assumptions on $\mathcal{R}_i$. The Regret Theorem provides a far-reaching new generalization of a well known device from [6] for facilitating the solution of optimization problems (1.2) when $\mathcal{R}_i$ is a CVaR risk measure. The Regression Theorem in Section 5 handles problem (1.4) on level beyond anything previously attempted.

Duality will occupy our attention in Section 6. Each of the quantifiers $\mathcal{R}$, $\mathcal{D}$, $\mathcal{E}$, $\mathcal{V}$, has a dual expression in the presence of "closed convexity", a property we will build into them in Section 3. This is presented in the Envelope Theorem. Such dualizations shed additional light on modeling motivations. Although the dualization of a risk measure $\mathcal{R}$ has already been closely investigated, its advantageous coordination with the dualization of $\mathcal{V}$ is new here together with its echoes in $\mathcal{D}$ and $\mathcal{E}$.

Expectation quadrangles. Many examples, but by no means all, will fall into the category that we call the *expectation case* of the risk quadrangle. The special feature in this case is that

$$\mathcal{E}(X) = E[e(X)], \quad \mathcal{V}(X) = E[v(X)], \quad \mathcal{U}(Y) = E[u(Y)], \tag{1.9}$$

for functions e and v on $(-\infty, \infty)$ related to each other by

$$e(x) = v(x) - x, \quad v(x) = e(x) + x, \tag{1.10}$$

and on the other hand, v corresponding to relative utility u through

$$v(x) = -u(-x), \quad u(y) = -v(-y). \tag{1.11}$$

The $\mathcal{V} \leftrightarrow \mathcal{E}$ correspondence in Diagram 3 holds under (1.9), while (1.10) ensures that $\mathcal{V}(X) = -\mathcal{U}(-X)$ and $\mathcal{U}(Y) = -\mathcal{V}(-Y)$. The consequences for the $\mathcal{S}$, $\mathcal{R}$ and $\mathcal{D}$ components of the quadrangle, as generated by the other formulas in Diagram 3, will be discussed in Section 4.

Expected utility is a central notion in decision analysis in economics and likewise in finance, cf. [10]. Expected error expressions similarly dominate much of statistics, cf. [11]. Expectation quadrangles provide the connection to those bodies of theory in the development undertaken here. However, the quadrangle scheme also reveals serious limitations of the expectation case. Many attractive examples do not fit into it, as will be clear in the sampling of Section 2. Even expressions $\mathcal{U}(Y) = E[u_0(Y - Y_0)]$, giving expected

[11] See the Regression Theorem in Section 5. In general, $\mathcal{S}$ can assign an interval of values, so the constraint in (1.5) would better be written as $0 \in \mathcal{S}(Z_f)$.

[12] Other issues in statistical "estimation", such as the convergence of approximations based on sampling are not taken up here despite their great importance in the long run. This is due to the lack of space and, in some cases, the imperfect state of current knowledge. Interesting research challenges abound.

u_0-utility relative to a benchmark gain Y_0, can fail to be directly representable as $\mathcal{U}(Y) = E[u(Y)]$ for a utility function u. Departure from expected utility and expected error is therefore inevitable, if the quadrangle relationships we are exploring are to reach their full potential for application. This widening of perspective is another of the contributions we are aiming at here.

2. Some examples showing the breadth of the scheme

Before going into technical details, we will look at an array of examples aimed at illustrating the scope and richness of the quadrangle scheme and the interrelationships it reveals. In each case the elements correspond to each other in the manner of Diagram 3. Some of the connections are already known but have not all been placed in a single, comprehensive picture.

The first example ties classical safety margins in the risk measure format in optimization and reliability engineering to the standard tools of least-squares regression. It centers on the mean value of X as the statistic. The scaling factor $\lambda > 0$ allows the safety margin to come into full play: having X adequately $\leq C$ is interpreted as having $\mu(X)$ at least λ standard deviation units $\leq C$. Through "regret" a link is made to an associated "utility". However, as will be seen in Section 3, this quadrangle lacks an important property of "coherency".

Example 1 (*A Mean-based Quadrangle (With $\lambda > 0$ as a Scaling Parameter)*)**.**

$\mathcal{S}(X) = EX = \mu(X) =$ mean
$\mathcal{R}(X) = \mu(X) + \lambda\sigma(X) =$ safety margin tail risk
$\mathcal{D}(X) = \lambda\sigma(X) =$ standard deviation, scaled
$\mathcal{V}(X) = \mu(X) + \lambda\|X\|_2 = L^2$-regret, scaled
$\mathcal{E}(X) = \lambda\|X\|_2 = L^2$-error, scaled.

Regression with this $\mathcal{E}$ corresponds through (1.5) to minimizing the standard deviation of the error $Z_f = Y - f(X_1, \ldots, X_n)$ subject to the mean of the error being 0.

Already here we have an example that is not an expectation quadrangle. Perhaps that may seem a bit artificial, because the $\mathcal{L}^2$-norm could be replaced by its square. That would produce a modified quadrangle giving the same statistic:

Example 1′ (*Variance Version of* Example 1)**.**

$\mathcal{S}(X) = EX = \mu(X)$
$\mathcal{R}(X) = \mu(X) + \lambda\sigma^2(X)$
$\mathcal{D}(X) = \lambda\sigma^2(X)$
$\mathcal{V}(X) = \mu(X) + \lambda\|X\|_2^2 = E[v(X)]$ for $v(x) = x + \lambda x^2$
$\mathcal{E}(X) = \lambda\|X\|_2^2 = E[e(x)]$ for $e(x) = \lambda x^2$.

However, some properties would definitely change. The first version has $\mathcal{R}(X + X') \leq \mathcal{R}(X) + \mathcal{R}(X')$, which is a rule often promoted for measures of risk as part of "coherency" (as explained in Section 3), but this fails for the second version (although "convexity" persists). A new quadrangle variant of Examples 1 and 1′ with potentially important advantages will be introduced in Example 7.

The next example combines quantile statistics with concepts coming from risk management in finance and engineering. By tying "conditional value-at-risk", on the optimization side, to quantile regression (in contrast to least-squares regression) as pioneered in statistics by Koenker and Bassett [12], it underscores a unity that might go unrecognized without the risk quadrangle scheme.

The key in this case is provided by the (cumulative) distribution function $F_X(x) = \text{prob}\{X \leq x\}$ of a random variable X and the quantile values associated with it. If, for a probability level $\alpha \in (0, 1)$, there is a unique x such that $F_X(x) = \alpha$, that x is the α-quantile $q_\alpha(X)$. In general, however, there are two values to consider as extremes:

$$q_\alpha^+(X) = \inf\{x \mid F_X(x) > \alpha\},$$
$$q_\alpha^-(X) = \sup\{x \mid F_X(x) < \alpha\}. \tag{2.1}$$

It is customary, when these differ, to take the lower value as "the" α-quantile, noting that, because F_X is right-continuous, this is the lowest x such that $F_X(x) = \alpha$. Here, instead, we will consider the entire *interval* between the two competing values as the quantile,

$$q_\alpha(X) = [q_\alpha^-(X), q_\alpha^+(X)], \tag{2.2}$$

bearing in mind that this interval usually collapses to a single value. That approach will fit better with our way of defining a "statistic" by the argmin notation. Also important to understand, in our context of interpreting X as a "cost" or "loss", is that the notion of *value-at-risk* in finance coincides with quantile. There is an upper value-at-risk $\text{VaR}_\alpha^+(X) = q_\alpha^+(X)$ along with a lower value-at-risk $\text{VaR}_\alpha^-(X) = q_\alpha^-(X)$, and, in general, a value-at-risk *interval* $\text{VaR}_\alpha(X) = [\text{VaR}_\alpha^+(X), \text{VaR}_\alpha^-(X)]$ identical to the quantile interval $q_\alpha(X)$.

Besides value-at-risk, the example coming under consideration involves the *conditional value-at-risk* of X at level $\alpha \in (0, 1)$ as defined by

$$\text{CVaR}_\alpha(X) = \text{expectation of } X \text{ in its } \alpha\text{-tail}, \tag{2.3}$$

which is also expressible by

$$\text{CVaR}_\alpha(X) = \frac{1}{1-\alpha}\int_\alpha^1 \text{VaR}_\tau(X)\,d\tau. \tag{2.4}$$

The second formula is due to Acerbi [13] in different terminology, while the first follows the pattern in [5], where "conditional value-at-risk" was coined.[13] Due to applications of risk theory in areas outside of finance, such as reliability engineering, we believe it is advantageous to maintain, parallel to value-at-risk and quantile, the ability to refer to the conditional value-at-risk $\text{CVaR}_\alpha(X)$ equally as the *superquantile* $\bar{q}_\alpha(X)$. We will be helped here and later by the notation

$$X = X_+ - X_- \quad \text{with } X_+ = \max\{0, X\}, X_- = \max\{0, -X\}.$$

Example 2 (*A Quantile-based Quadrangle (at Any Confidence Level $\alpha \in (0, 1)$)*)**.**

$\mathcal{S}(X) = \text{VaR}_\alpha(X) = q_\alpha(X) =$ quantile
$\mathcal{R}(X) = \text{CVaR}_\alpha(X) = \bar{q}_\alpha(X) =$ superquantile
$\mathcal{D}(X) = \text{CVaR}_\alpha(X - EX) = \bar{q}_\alpha(X - EX)$
$\quad = $ superquantile-deviation

$\mathcal{V}(X) = \dfrac{1}{1-\alpha}EX_+ =$ average absolute loss, scaled[14]

$\mathcal{E}(X) = E\left[\dfrac{\alpha}{1-\alpha}X_+ + X_-\right]$
$\quad =$ normalized Koenker–Bassett error.

This is an expectation quadrangle with

$$e(x) = \frac{\alpha}{1-\alpha}\max\{0, x\} + \max\{0, -x\},$$
$$v(x) = \frac{1}{1-\alpha}\max\{0, x\}, \qquad u(y) = \frac{1}{1-\alpha}\min\{0, y\}.$$

[13] The α-tail distribution of X corresponds to the upper part of the distribution of X having probability $1 - \alpha$. The interpretation of this for the case when F_X has a jump at the α quantile is worked out in [6].

[14] Average absolute loss as "regret", and as inspiration for the terminology we are introducing here more broadly, goes back to [14] in stochastic programming.

The original Koenker–Bassett error expression differs from the one here by a positive factor. Adjustment is needed to make it project to the desired $\mathcal{D}$. With respect to this measure of error, regression has the interpretation in (1.5) that the α-superquantile (or α-CVaR) deviation of Z_f is minimized subject to the α-quantile of Z_f being 0.

The targeting of average loss as the source of "regret" in Example 2 is interesting because of the role that average loss has long had in stochastic optimization, but also through the scaling feature. In the past, such scaling might have been thought immaterial, but this quadrangle shows that it identifies a particular loss quantile having a special role.

Example 2 confirms the motivations in Section 1 for looking at entire quadrangles. Consider a stochastic optimization problem in the form of (1.2). It is tempting, and common in many applications, to contemplate taking $\mathcal{R}_i$ to be a quantile q_{α_i}. The constraint $\mathcal{R}_i(X_i(x)) \le c_i$ would require then that x be chosen so that the random "cost" $X_i(x)$ is $\le c_i$ with probability at least α_i. However, this apparently natural approach suffers from the fact that $q_{\alpha_i}(X_i(x))$ may be poorly behaved as a function of x as well as subject to the indeterminacy, or discontinuity, associated with (2.2). That could hamper computation and lead to instability of solutions.

An alternative to a quantile would be to take $\mathcal{R}_i$ to be a superquantile $\bar{q}_{\alpha_i}$. The constraint $\mathcal{R}_i(X_i(x)) \le c_i$, as an expression of $X_i(x)$ being "adequately" $\le c_i$, is then more conservative and has an interpretation in terms of "buffered probability of failure", cf. [15]. Moreover it is better behaved and able to preserve convexity of $X_i(x)$ with respect to x, if present. A further advantage in optimization from such an approach is seen from the projection from $\mathcal{V}$ to $\mathcal{R}$ on the left side of the quadrangle:

$$\bar{q}_{\alpha_i}(X_i(x)) \le c_i \iff C_i + \frac{1}{1-\alpha_i} E[\max\{0, X_i(x) - C_i\}] \le c_i \quad \text{for a choice of } C_i \in \mathbb{R}.$$

Thus, a superquantile (or CVaR) constraint can be reformulated as something simpler through the introduction of *another decision variable* C_i alongside of x.[15] In some situations the expectation term in the reformulation can even be handled through linear programming. This first came out in [5], but the point to be emphasized here is that such a device is not limited to superquantiles. The same effect can be achieved with a risk measure $\mathcal{R}_i$ and regret measure $\mathcal{V}_i$ pair from *any* quadrangle (with "regularity"), replacing $\mathcal{R}_i(X_i(x)) \le c_i$ by $C_i + \mathcal{V}_i(X_i(x) - C_i) \le c_i$, and the variable C_i ends up then in optimality as $\mathcal{S}_i(X_i(x))$; see the Regret Theorem in Section 5.

The fact that the $\mathcal{D}$–$\mathcal{E}$ side of the quadrangle in Example 2 corresponds to quantile regression has important implications as well. It was explained in Section 1 that factor models might be employed to replace $X_i(x)$ by some $\hat{X}_i(x)$ through regression, and that evidence suggests selecting for this regression the error measure $\mathcal{E}_i$ in the same quadrangle as the risk measure $\mathcal{R}_i$. It follows that, in an optimization problem (1.2) with objective and constraints of superquantile/CVaR type, quantile regression is perhaps most appropriate, at least in some linear models,[16] and should even be carried out at the α_i threshold chosen for each i.[17] Another observation is that quantile regression at level α_i turns into minimization of the superquantile/CVaR deviation measure $\mathcal{D}_{\alpha_i}(X) = \bar{q}_{\alpha_i}(X - EX)$ for $X = Z_f$ in (1.4). This is laid out in general by the Regression Theorem of Section 5. Only the quadrangle scheme is capable of bringing all this together.

[15] Minimizing $q_{\alpha_0}(X_0(x))$ in x converts likewise to minimizing $C_0 + \frac{1}{1-\alpha_0} E[\max\{0, X_0(x) - C_0\}]$ in x and C_0.

[16] This is a fertile topic for more research.

[17] Again, this is an insight applicable not only to Example 2, but to any of the other quadrangles that will come up.

The special case of Example 2 in which the quantile is the median is worth looking at directly. It corresponds to the error measure being the $\mathcal{L}^1$-norm in contrast to Example 1, where the error measure was the $\mathcal{L}^2$-norm. This furnishes a statistical alternative to least-squares regression in which the mean is replaced by the median, which may in some situations be a better way of centering a random cost.[18] Regression comes out then in (1.5) as minimizing the mean absolute deviation of the error random variable Z_f subject to it having 0 as its median.

Example 3 (*A Median-based Quadrangle (the Quantile Case for* $\alpha = \frac{1}{2}$)).

$$\mathcal{S}(X) = \mathrm{VaR}_{1/2}(X) = q_{1/2}(X) = \text{median}$$
$$\mathcal{R}(X) = \mathrm{CVaR}_{1/2}(X) = \bar{q}_{1/2}(X) = \text{"supermedian" (average in median-tail)}$$
$$\mathcal{D}(X) = E|X - q_{1/2}(X)| = \text{mean absolute deviation}$$
$$\mathcal{V}(X) = 2EX_+ = \mathcal{L}^1\text{-regret}$$
$$\mathcal{E}(X) = E|X| = \mathcal{L}^1\text{-error}.$$

For the sake of comparison, it is instructive to ask what happens if the error measure $\mathcal{E}$ on the estimation side, and in potential application to generalized regression, is the $\mathcal{L}^\infty$-norm. This leads to our fourth example, which emphasizes the case where X is (essentially) bounded.

Example 4 (*A Range-based Quadrangle (with* $\lambda > 0$ *as a Scaling Parameter)*).

$$\mathcal{S}(X) = \frac{1}{2}[\sup X + \inf X] = \text{center of range of } X \text{ (if bounded)}$$
$$\mathcal{R}(X) = EX + \frac{\lambda}{2}[\sup X - \inf X] = \text{range-buffered risk, scaled}$$
$$\mathcal{D}(X) = \frac{\lambda}{2}[\sup X - \inf X] = \text{radius of the range of } X \text{ (maybe } \infty\text{), scaled}$$
$$\mathcal{V}(X) = EX + \lambda \sup|X| = \mathcal{L}^\infty\text{-regret, scaled}$$
$$\mathcal{E}(X) = \lambda \sup|X| = \mathcal{L}^\infty\text{-error, scaled}.$$

This is *not* an expectation quadrangle. Having X adequately $\le C$ means here that X is kept below C by a margin equal to λ times the radius of the range of X. The interpretation of regression provided by (1.5) is that the radius of the range of Z_f is minimized subject to its center being at 0.

The example offered next identifies both as the statistic and as the risk the "worst cost" associated with X. It can be regarded as the limit of the quantile-based quadrangle in Example 2 as $\alpha \to 1$.

Example 5 (*A Worst-case-based Quadrangle*).

$$\mathcal{S}(X) = \sup X = \text{top of the range of } X \text{ (maybe } \infty\text{)}$$
$$\mathcal{R}(X) = \sup X = \text{yes, the same as } \mathcal{S}(X)$$
$$\mathcal{D}(X) = \sup X - EX = \text{span of the upper range of } X \text{ (maybe } \infty\text{)}$$
$$\mathcal{V}(X) = \begin{cases} 0 & \text{if } X \le 0 \\ \infty & \text{if } X \not\le 0 \end{cases} = \text{worst-case-regret}$$
$$\mathcal{E}(X) = \begin{cases} E|X| & \text{if } X \le 0 \\ \infty & \text{if } X \not\le 0 \end{cases} = \text{worst-case-error}.$$

This is another expectation quadrangle but with functions of unusual appearance:

[18] Furthermore, this quadrangle and the other quantile quadrangles in Example 2 will be seen to exhibit the "coherency" that was lacking in Example 1, and for that matter, Example 1′.

$$e(x) = \begin{cases} -x & \text{if } x \le 0 \\ \infty & \text{if } x > 0, \end{cases} \qquad v(x) = \begin{cases} 0 & \text{if } x \le 0 \\ \infty & \text{if } x > 0, \end{cases}$$

$$u(y) = \begin{cases} -\infty & \text{if } y < 0 \\ 0 & \text{if } y \ge 0. \end{cases}$$

The "range" of X here is its essential range, i.e., the smallest closed interval in which outcomes must lie with probability 1. Thus, in Example 5, the inequality $\mathcal{R}(X) \le C$ gives the risk-measure code for insisting that $X \le C$ with probability 1. The regret measure $\mathcal{V}(X)$ assigns infinite penalty when this is violated, but no disincentive otherwise. The regression associated with the quadrangle in Example 5 is one-sided. It corresponds in (1.5) to minimizing $|EZ_f|$ subject to $\sup Z_f = 0$.

A major attraction of the risk measure in Example 5 is that, on the surface at least, it apparently bypasses having to think about probabilities. This is the central theme of so-called "robust optimization".[19] However, a generalization can be made in which some additional probabilistic insights are available, and the appraisal of "worst" is distributed over different visions of the future tied to a coarse level of probability modeling. The details will not be fully understandable until we begin posing risk in the rigorous framework of a probability space in Section 3 (and all the more in Section 6), but we proceed anyway here to a suggestive preliminary formulation. It depends on partitioning the underlying uncertainty about the future into several different "sets of circumstances" $k = 1, \dots, r$ having no overlap[20] and letting

$$\begin{aligned} &p_k = \text{probability of the } k\text{th set of circumstances,} \\ &\quad \text{with } p_k > 0, \qquad p_1 + \cdots + p_r = 1, \\ &\sup_k X = \text{worst of } X \text{ under circumstances } k, \\ &\quad \text{for } k = 1, \dots, r, \\ &E_k X = \text{conditional expectation of } X \text{ under circumstances } k. \end{aligned} \tag{2.5}$$

The last implies, of course, that $p_1 E_1 X + \cdots + p_r E_r X = EX$.

Example 6 (*A Distributed-worst-case-based Quadrangle (With Respect to (2.5))*)**.**

$$\begin{aligned} &\mathcal{S}(X) = p_1 \sup_1 X + \cdots + p_r \sup_r X \\ &\mathcal{R}(X) = p_1 \sup_1 X + \cdots + p_r \sup_r X = \text{yes, the same as } \mathcal{S}(X) \\ &\mathcal{D}(X) = p_1[\sup_1 X - E_1 X] + \cdots + p_r[\sup_r X - E_r X] \\ &\mathcal{V}(X) = \begin{cases} 0 & \text{if } p_1 \sup_1 X + \cdots + p_r \sup_r X \le 0, \\ \infty & \text{if } p_1 \sup_1 X + \cdots + p_r \sup_r X > 0 \end{cases} \\ &\mathcal{E}(X) = \begin{cases} E|p_1 \sup_1 X + \cdots + p_r \sup_r X| \\ \quad \text{if } p_1 \sup_1 X + \cdots + p_r \sup_r X \le 0, \\ \infty \quad \text{if } p_1 \sup_1 X + \cdots + p_r \sup_r X > 0. \end{cases} \end{aligned}$$

This novel example is again *not* an expectation quadrangle. Moreover, unlike the previous cases, the quantifiers in Example 6 are not "law-invariant", i.e., their effects on X depend on more than just the distribution function F_X. It should be noted that expectations only enter the elements on the right side of this quadrangle. As far as optimization is concerned, by itself, there are no assumptions about probability structure other than the first line of (2.5). This can be regarded as a compromise between the starkness of Example 5 and a full-scale probability model.

We turn now to an example motivated especially from the estimation side. It concerns an expectation quadrangle which interpolates between Examples 1′ and 3 by looking at an error expression like the one in Huber's modification of least-squares regression in order to mollify the influence of outliers. We introduce a scaling parameter $\beta > 0$ and make use of the β-*truncation* function

$$T_\beta(x) = \begin{cases} \beta & \text{when } x \ge \beta, \\ x & \text{when } -\beta \le x \le \beta, \\ -\beta & \text{when } x \le -\beta. \end{cases}$$

Example 7 (*A Truncated-mean-based Quadrangle (with Scaling Parameter $\beta > 0$)*)**.**

$$\begin{aligned} &\mathcal{S}(X) = \mu_\beta(X) = \text{value of } C \text{ such that } E[T_\beta(X - C)] = 0 \\ &\mathcal{R}(X) = \mu_\beta(X) + E[v(X - \mu_\beta(X))] \quad \text{for } v \text{ as below} \\ &\mathcal{D}(X) = E[e(X - \mu_\beta(X))] \quad \text{for } e \text{ as below} \\ &\mathcal{V}(X) = E[v(X)] \quad \text{with } v(x) = \begin{cases} -\dfrac{\beta}{2} & \text{when } x \le -\beta \\ x + \dfrac{1}{2\beta}x^2 & \text{when } |x| \le \beta \\ 2x - \dfrac{\beta}{2} & \text{when } x \ge \beta \end{cases} \\ &\mathcal{E}(X) = E[e(X)] \quad \text{with } e(x) = \begin{cases} |x| - \dfrac{\beta}{2} & \text{when } |x| \ge \beta \\ \dfrac{1}{2\beta}x^2 & \text{when } |x| \le \beta \end{cases} \\ &\qquad = \text{Huber-type error.} \end{aligned}$$

In the limit of $\mu_\beta(X)$ as $\beta \to \infty$, we end up with just EX, as in Examples 1 and 1′. For the deviation measure $\mathcal{D}$ in Example 7, one can think of $2\beta\mathcal{D}(X)$ as the β-truncation $\sigma^2_\beta(X)$ of $\sigma^2(X)$. It approaches that variance as $\beta \to \infty$. In the corresponding regression, interpreted through (1.5), $\sigma^2_\beta(Z_f)$ is minimized subject to $\mu_\beta(Z_f) = 0$ for the error random variable Z_f. This contrasts with minimizing $\sigma^2(Z_f)$ subject to $\mu(Z_f) = 0$ in Examples 1 and 1′.

As a quadrangle, Example 7 is brand new. Its noteworthy feature, as contrasted with the limiting case in Example 1′, is that its $v(x)$ is a *nondecreasing* convex function of x.[21] In consequence, $\mathcal{V}$ and $\mathcal{R}$ will be "monotonic" (as defined in Section 3) and their dualizations (in Section 6) will fit into a framework of probability which the dualizations coming out of Example 1′ cannot attain.

The next quadrangle, again in the expectation case, looks very different. The log-exponential risk measure at the heart of it is a recognized tool in risk theory in finance,[22] but its connection with a form of generalized regression, by way of the $\mathcal{D}$–$\mathcal{E}$ side of the quadrangle, has not previously been contemplated. As in Examples 5 and 6, the risk $\mathcal{R}(X)$ equals the statistic $\mathcal{S}(X)$.

Example 8 (*A Log-Exponential-Based Quadrangle*)**.**

$$\begin{aligned} &\mathcal{S}(X) = \log E[\exp X] \\ &\qquad = \text{expression dual to Boltzmann–Shannon entropy}[23] \\ &\mathcal{R}(X) = \log E[\exp X] = \text{yes, the same as } \mathcal{S}(X) \\ &\mathcal{D}(X) = \log E[\exp(X - EX)] = \text{log-exponential deviation} \\ &\mathcal{V}(X) = E[\exp X - 1] = \text{exponential regret} \longleftrightarrow \mathcal{U}(Y) \\ &\qquad = E[1 - \exp(-Y)] \\ &\mathcal{E}(X) = E[\exp X - X - 1] = \text{(unsymmetric) exponential error.} \end{aligned}$$

[19] In that subject, probabilistic assessments typically enter nevertheless through construction of an "uncertainty set" consisting of the future states or scenarios deemed worthy of consideration in the worst-case analysis. That uncertainty set can be identified as the Ω set in the probability-space underpinnings of risk theory explained in Section 3.

[20] Technically this refers to "events" as measurable subsets Ω_k of the probability space Ω introduced in the next section. For more, see also (6.10) and what precedes it. In the context of "robust optimization", one can think of the chosen "uncertainty set" Ω being partitioned into a number of smaller uncertainty sets Ω_k to which relative probabilities can be assigned. By admitting various degrees of fineness in the partitioning (fields of sets providing "filters of information"), a bridge is provided between different layers of probability knowledge.

[21] And the associated utility $u(y)$ will be a *nondecreasing* concave function of y.

[22] This is called *entropic risk* in [10].

[23] The "exp" notation is adopted so as not to conflict with the convenient use of "e" for error integrands in (1.9).

Regression here can be interpreted through (1.5) as minimizing $\log E[\exp(Z_f - EZ_f)] = \log E[\exp Z_f] - EZ_f$ subject to $\log E[\exp Z_f] = 0$, or equivalently minimizing $|EZ_f|$ subject to $E[\exp Z_f] = 1$ (since the latter implies $\exp EZ_f \le 1$, hence $EZ_f < 0$).

The regret $\mathcal{V}$ in Example 8 is paired with an expected utility expression that is commonly employed in finance: we are in the expectation case with

$$e(x) = \exp x - x - 1, \qquad v(x) = \exp x - 1,$$
$$u(y) = 1 - \exp(-y).$$

Such utility pairing is seen also in the coming Example 9, which fits the expectation case with

$$e(x) = \begin{cases} \log \dfrac{1}{1-x} - x & \text{if } x < 1 \\ \infty & \text{if } x \ge 1, \end{cases}$$
$$v(x) = \begin{cases} \log \dfrac{1}{1-x} & \text{if } x < 1 \\ \infty & \text{if } x \ge 1, \end{cases}$$
$$u(y) = \begin{cases} \log(1+y) & \text{if } y > -1 \\ -\infty & \text{if } y \le -1. \end{cases}$$

Example 9 (*A Rate-Based Quadrangle*)**.**

$$\mathcal{S}(X) = r(X) = \text{unique } C \ge \sup X - 1$$
$$\text{such that } E\left[\frac{1}{1-X+C}\right] = 1$$
$$\mathcal{R}(X) = r(X) + E\left[\log \frac{1}{1-X+r(X)}\right]$$
$$\mathcal{D}(X) = r(X) + E\left[\log \frac{1}{1-X+r(X)} - X\right]$$
$$\mathcal{V}(X) = E\left[\log \frac{1}{1-X}\right] \longleftrightarrow \mathcal{U}(Y) = E[\log(1+Y)]$$
$$\mathcal{E}(X) = E\left[\log \frac{1}{1-X} - X\right].$$

We have dubbed this quadrangle "rate-based" because, in the utility connection, $\log(1+y)$ is an expression applied to a rate of gain y (which of necessity is > -1); cf. [16, Chapter 15], for the role of this in finance. Correspondingly in $\log \frac{1}{1-x}$, we are dealing with a rate of loss.

The next two examples in this section lie again outside the expectation case and present a more complicated picture where error and regret are defined by an auxiliary operation of minimization. The first concerns "mixed" quantiles/VaR and superquantiles/CVaR. The idea, from the risk measure perspective, is to study expressions of the type

$$\mathcal{R}(X) = \int_0^1 \text{CVaR}_\alpha(X)\, d\lambda(\alpha) \tag{2.6}$$

for any weighting measure λ on $(0, 1)$ (nonnegative with total measure 1). In particular, if λ is comprised of atoms with weights $\lambda_k > 0$ at points α_k for $k = 1, \ldots, r$, with $\lambda_1 + \cdots + \lambda_r = 1$, one gets

$$\mathcal{R}(X) = \lambda_1 \text{CVaR}_{\alpha_1}(X) + \cdots + \lambda_r \text{CVaR}_{\alpha_r}(X). \tag{2.7}$$

The question is whether this can be placed in a full quadrangle in the format of Diagrams 1–3.

Incentive comes from the fact that such risk measures have a representation as "spectral measures" in the sense of [13], which capture preferences in terms of "risk profiles".[24] We proved in [21, Proposition 5] (echoing our working paper [22]) that, as long as the weighting measure λ satisfies $\int_0^1 (1-\alpha)^{-1} d\lambda(\alpha) < \infty$, the risk measure in (2.7) can equivalently be expressed in the form

$$\mathcal{R}(X) = \int_0^1 \text{VaR}_\tau(X)\phi(\tau)\, d\tau$$
$$\text{with } \phi(\tau) = \int_{(0,\tau]} (1-\alpha)^{-1} d\lambda(\alpha), \tag{2.8}$$

where the function ϕ, defined on $(0, 1)$, gives the *risk profile*.[25]

The risk profile for a single "unmixed" risk measure CVaR_α is the function ϕ_α that has the value $1/(1-\alpha)$ on $[\alpha, \infty)$ but 0 on $(0, \alpha)$; this corresponds to formula (2.4). Moreover the risk profile for a weighted CVaR sum as in (2.7) would be the step function $\phi = \lambda_1 \phi_{\alpha_1} + \cdots + \lambda_r \phi_{\alpha_r}$.

Although the quadrangle that would serve for a general weighting measure in (2.6) is still a topic of research, the special case in (2.7) is accessible from the platform of [9], which will be widened in Section 4 (in the Mixing Theorem).

Example 10 (*A Mixed-Quantile-based Quadrangle (For any Confidence Levels $\alpha_i \in (0, 1)$ and Weights $\lambda_k > 0$, $\sum_{k=1}^r \lambda_k = 1$)*)**.**

$$\begin{aligned} \mathcal{S}(X) &= \sum_{k=1}^r \lambda_k q_{\alpha_k}(X) = \sum_{k=1}^r \lambda_k \text{VaR}_{\alpha_k}(X) \\ &= \text{a mixed quantile}[26] \end{aligned}$$

$$\begin{aligned} \mathcal{R}(X) &= \sum_{k=1}^r \lambda_k \bar{q}_{\alpha_k}(X) = \sum_{k=1}^r \lambda_k \text{CVaR}_{\alpha_k}(X) \\ &= \text{a mixed superquantile} \end{aligned}$$

$$\begin{aligned} \mathcal{D}(X) &= \sum_{k=1}^r \lambda_k \bar{q}_{\alpha_k}(X - EX) \\ &= \sum_{k=1}^r \lambda_k \text{CVaR}_{\alpha_k}(X - EX) \\ &= \text{the corresponding mixture of} \\ &\quad \text{superquantile deviations} \end{aligned}$$

$$\begin{aligned} \mathcal{V}(X) &= \min_{B_1,\ldots,B_r} \left\{ \sum_{k=1}^r \lambda_k \mathcal{V}_{\alpha_k}(X - B_k) \mid \sum_{k=1}^r \lambda_k B_k = 0 \right\} \\ &= \text{a derived balance of the regrets} \end{aligned}$$
$$\mathcal{V}_{\alpha_k}(X) = \frac{1}{1-\alpha_k} EX_+$$

$$\begin{aligned} \mathcal{E}(X) &= \min_{B_1,\ldots,B_r} \left\{ \sum_{k=1}^r \lambda_k \mathcal{E}_{\alpha_k}(X - B_k) \mid \sum_{k=1}^r \lambda_k B_k = 0 \right\} \\ &= \text{a derived balance of the errors} \end{aligned}$$
$$\mathcal{E}_{\alpha_k}(X) = E\left[\frac{\alpha_k}{1-\alpha_k} X_+ + X_-\right].$$

[24] Such profiles occur in "dual utility theory", a subject addressed by Yaari [17] and Roell [18] and recently revisited with greater rigor by Dentcheva and Ruszczyński [19]. Their integrals are the "concave distortion" functions seen in finance and insurance theory, cf. [10,20].

[25] This function ϕ is right-continuous and nondecreasing with $\phi(0^+) = 0$, $\phi(1^-) < \infty$ and $\int_0^1 \phi(\tau)\, d\tau = 1$. Conversely, any function ϕ with those properties arises from a unique choice of λ as described. The cited sources have a reversed formula due to X being gain-oriented instead of loss-oriented, as here.

[26] This kind of sum, in which some of the terms could be intervals, is to be interpreted in general as referring to all results obtained by selecting particular values within those intervals.

The case of a general weighting measure may be approximated this way arbitrarily closely, as can very well be seen through the corresponding risk profiles. When the measure is concentrated in finitely many points, the corresponding profile function ϕ in (2.8) is a step function, and vice versa, as already noted. An arbitrary profile function ϕ (fulfilling the conditions indicated above in a footnote) can be approximated by a profile function that is a step function.

A highly interesting use for the quadrangle of Example 10 is the *mixed quantile approximation of a superquantile*. According to (2.4), the value $\bar{q}_\alpha(X) = \mathrm{CVaR}_\alpha(X)$ can be obtained by calculating the integral of $q_\tau(X) = \mathrm{VaR}_\tau(X)$ over $\tau \in [\alpha, 1]$. Classical numerical approaches introduce a finite subdivision of the interval $[\alpha, 1]$ and replace the integrand by a nearby step function or piecewise linear function based on the quantiles marking that subdivision. It is easy to see that the value of the integral for that approximated integrand is actually a mixed quantile expression. The conclusion is that versions of the quadrangle of Example 10 can serve as approximations to a *superquantile-based quadrangle* parallel to the quantile-based quadrangle of Example 2.[27] In this manner, *superquantile regression*, in which the statistic is a superquantile instead of a quantile, can be carried out.

Although the mixed superquantile/CVaR risk measures $\mathcal{R}$ in Example 10 have a well recognized importance in expressing preferences toward risk, through the profiles explained above,[28] the identification in this quadrangle of a corresponding "optimally mixed" regret measure $\mathcal{V}$ for such $\mathcal{R}$ is new. The associated error measure $\mathcal{E}$ is the one thereby indicated for use in regression approximations where this kind of risk measure is involved.

It is worth emphasizing that the min expressions for $\mathcal{V}$ and $\mathcal{E}$ in Example 10 are no impediment at all in practice when applied to optimization or regression. For instance, the trick explained after Example 2 for simplifying a superquantile/CVaR constraint through the introduction of an additional decision variable works here as well. The only difference is that still more decision variables corresponding to the B's in the quadrangle are introduced, too.

The following example likewise offers something new as far as risk measures and potential applications in regression are concerned, although the "statistic" in question has already come up in mortgage pipeline hedging; see [24].

Example 11 (*A Quantile-Radius-Based Quadrangle (For any $\alpha \in (1/2, 1)$ and $\lambda > 0$)*)**.**

$$\begin{aligned}
\mathcal{S}(X) &= \frac{1}{2}[q_\alpha(X) - q_{1-\alpha}(X)] = \frac{1}{2}[\mathrm{VaR}_\alpha(X) - \mathrm{VaR}_{1-\alpha}(X)] \\
&= \text{the } \alpha\text{-quantile radius of } X, \text{ or } \frac{1}{2}\text{-two-tail-VaR}_\alpha \text{ of } X \\
\mathcal{R}(X) &= EX + \frac{\lambda}{2}[\bar{q}_\alpha(X) + \bar{q}_\alpha(-X)] \\
&= EX + \frac{\lambda}{2}[\mathrm{CVaR}_\alpha(X) + \mathrm{CVaR}_\alpha(-X)] \\
&= \text{reverted CVaR}_\alpha, \text{ scaled} \\
\mathcal{D}(X) &= \frac{\lambda}{2}[\bar{q}_\alpha(X) + \bar{q}_\alpha(-X)] \\
&= \frac{\lambda}{2}[\mathrm{CVaR}_\alpha(X) + \mathrm{CVaR}_\alpha(-X)] \\
&= \text{the } \alpha\text{-superquantile radius of } X, \text{ scaled} \\
\mathcal{V}(X) &= EX + \min_B \left\{ \frac{\lambda}{2(1-\alpha)} E\,[[B + X]_+ + [B - X]_+] - B \right\} \\
&= \alpha\text{-quantile-radius regret in } X, \text{ scaled} \\
\mathcal{E}(X) &= \frac{\lambda}{2(1-\alpha)} \min_B E\,[[B + X]_+ + [B - X]_+] \\
&= \alpha\text{-quantile-radius error in } X, \text{ scaled.}
\end{aligned}$$

This example will be justified and extended in Section 3 (through the Reverting Theorem).

As the final example in this section, we offer a generalization of Example 2 to the "higher-order moment risk measures" introduced in [8] and further analyzed recently in [25]. The "quantile" terminology does not come from those works and is only imposed here in suggestion of the strong parallels with the earlier quantile-based quadrangle, which would be the case where $p = 1$.

Example 12 (*A Higher-Order Quantile-Based Quadrangle (For $\alpha \in (0, 1)$, $p \in (1, \infty)$)*)**.**

$$\begin{aligned}
&\mathcal{S}(X) = q_\alpha^{(p)}(X) = p\text{-moment quantile} \\
&\mathcal{R}(X) = \bar{q}_\alpha^{(p)}(X) = p\text{-moment superquantile} \\
&\mathcal{D}(X) = \bar{q}_\alpha^{(p)}(X - EX) = p\text{-moment superquantile-deviation} \\
&\mathcal{V}(X) = \frac{1}{1-\alpha}\|X_+\|_p = p\text{-normed absolute loss, scaled} \\
&\mathcal{E}(X) = \frac{1}{1-\alpha}\|X_+\|_p - EX = p\text{-moment quantile error.}
\end{aligned}$$

The p-moment quantile $q_\alpha^{(p)}(X)$ is known to be characterized by the equation

$$(1-\alpha)^{p-1} = \|(X - q_\alpha^{(p)}(X))_+\|_{p-1} / \|(X - q_\alpha^{(p)}(X))_+\|_p.$$

For this and other properties, see [8].

What considerations have to be faced in constructing further quadrangle examples? For instance, is there a full quadrangle with $\mathcal{R}(X) = EX$, or with $\mathcal{R}(X) = \mathrm{VaR}_\alpha(X) = q_\alpha(X)$? The answer is yes in both cases, provided that $\mathrm{VaR}_\alpha(X)$ and $q_\alpha(X)$ (which can be intervals in our setting) are replaced by $\mathrm{VaR}_\alpha^-(X)$ and $q_\alpha^-(X)$, say, but the resulting quadrangles are "not interesting". For $\mathcal{R}(X) = EX$, we must have $\mathcal{D}(X) \equiv 0$ in accordance with Diagram 3. An associated measure of error would be $\mathcal{E}(X) = |EX|$, which is paired with $\mathcal{V}(X) = EX + |EX| = 2\max\{0, EX\}$. Then $\mathcal{S}(X) = EX$ and offers us nothing new.

For $\mathcal{R}(X) = \mathrm{VaR}_\alpha^-(X)$, on the other hand, we have $\mathcal{D}(X) = \mathrm{VaR}_\alpha^-(X - EX)$ and could take $\mathcal{E}(X) = \mathrm{VaR}_\alpha^-(X - EX) + |EX|$ and correspondingly $\mathcal{V}(X) = \mathrm{VaR}_\alpha^-(X) + 2\max\{0, EX\}$. However, then we merely have $\mathcal{S}(X) = EX$. Some different and more interesting $\mathcal{V}(X)$ might project onto $\mathcal{R}(X) = \mathrm{VaR}_\alpha^-(X)$ through the formula in Diagram 3, but this remains to be seen.[29]

In a similar vein, it might be wondered whether the expression $\mathrm{VaR}_\alpha(X) - \mathrm{VaR}_{1-\alpha}(X)$ appearing as the statistic of Example 10 could serve as the deviation measure $\mathcal{D}(X)$ in some quadrangle, since it is nonnegative and vanishes for constant X. Again the answer is yes, but perhaps only trivially.

Anyway, the most important guideline for additional quadrangle examples is that the quantifiers must fit with the descriptions in Diagram 2, which have yet to be fleshed out with appeals to specific mathematical properties. That is our task in the coming section. Those properties have to make sense in applications and lead to a sturdy methodology, and the real trouble with $\mathcal{R}(X) = EX$ and $\mathcal{R}(X) = \mathrm{VaR}_\alpha^-(X)$ as measures of risk is that they fall short of meeting such a standard. The Quadrangle Theorem of the coming Section 3, our central result, will therefore not apply to them.

[27] In [23], direct expressions for the elements of this quadrangle are produced.

[28] See also the theoretical developments in [10, Chapter 4.5].

[29] No claim is made about there being a *unique* $\mathcal{E}$ projecting onto some $\mathcal{D}$, or a *unique* $\mathcal{V}$ projecting onto some $\mathcal{R}$, and indeed that must not be hoped for. The real issue instead is that of determining an "natural" antecedent with valuable characteristics. For instance, any risk measure $\mathcal{R}$ can be projected from $\mathcal{V}(X) = \mathcal{R}(X) + \lambda|EX|$ and any deviation measure from $\mathcal{E}(X) = \mathcal{D}(X) + \lambda|EX|$ for arbitrary $\lambda > 0$, with the pointless consequence that $\mathcal{S}(X) = EX$.

3. The main properties and relationships

This section is devoted to laying a rigorous foundation for the elements of the risk quadrangle and their interconnections. It also furnishes tools for generating additional quadrangles from given ones.

In working with random variables we adopt the standard model in probability theory, which interprets them as functions on a probability space. Specifically, we suppose there is an underlying space Ω with elements ω standing for future states, or scenarios, along with a measure which assigns probabilities to various subsets of Ω. There is no loss of generality in this, but technicalities come in which we wish to avoid getting too occupied with at present.[30] Random variables from now on are functions $X : \Omega \to \mathbb{R}$, but we restrict attention to those for which $E[X^2] < \infty$, indicating this by $X \in \mathcal{L}^2(\Omega)$. Here E is the expectation with respect to the background probability measure on Ω.[31]

Any $X \in \mathcal{L}^2(\Omega)$ also has $E|X| < \infty$, so that EX is well defined and finite. Furthermore, the variance $\sigma^2(X) = E[X - EX]^2$ and its square root, the standard deviation $\sigma(X)$, are well defined and finite.[32] These expressions characterize the natural ("strong") convergence in $\mathcal{L}^2(\Omega)$ of a sequence of random variables X^k to a random variable X:

$$\begin{aligned} \mathcal{L}^2\text{-}\lim_{k\to\infty} X^k = X \iff & \lim_{k\to\infty} \|X^k - X\|_2 = 0 \\ \iff & \lim_{k\to\infty} E[X^k - X] = 0 \quad \text{and} \\ & \lim_{k\to\infty} \sigma(X^k - X) = 0. \end{aligned} \tag{3.1}$$

In many applications Ω may consist of finitely many elements ω, each having a positive probability weight. The choice of norm makes no difference then, because $\mathcal{L}^2(\Omega)$ is finite-dimensional.[33]

The quantifiers $\mathcal{R}$, $\mathcal{D}$, $\mathcal{V}$ and $\mathcal{E}$, all of which assign numerical values, *possibly including* $+\infty$,[34] to random variables X, are said to be "functionals" on $\mathcal{L}^2(\Omega)$. Some of the properties that come up may be shared, so it is expedient to state them in terms of a general functional $\mathcal{F} : \mathcal{L}^2(\Omega) \to (-\infty, \infty]$:

- $\mathcal{F}$ is *convex* if $\mathcal{F}((1-\tau)X + \tau X') \le (1-\tau)\mathcal{F}(X) + \tau\mathcal{F}(X')$ for all X, X', and $\tau \in (0, 1)$.[35]
- $\mathcal{F}$ is *positively homogeneous* if $\mathcal{F}(0) = 0$ and $\mathcal{F}(\lambda X) = \lambda\mathcal{F}(X)$ for all $\lambda \in (0, \infty)$.
- $\mathcal{F}$ is *subadditive* if $\mathcal{F}(X + X') \le \mathcal{F}(X) + \mathcal{F}(X')$ for all X, X'.
- $\mathcal{F}$ is *monotonic* (nondecreasing, here) if $\mathcal{F}(X) \le \mathcal{F}(X')$ when $X \le X'$.[36]
- $\mathcal{F}$ is *closed* if, for all $C \in \mathbb{R}$, the set $\{X \mid \mathcal{F}(X) \le C\}$ is closed.[37]

Convexity will be valuable for much of what we undertake. Positive homogeneity is a more special property which, in the study of risk, was emphasized more in the past than now. An elementary fact of convex analysis is that

$$\begin{aligned} &\mathcal{F} \text{ convex} + \text{positively homogeneous} \\ &\iff \mathcal{F} \text{ subadditive} + \text{positively homogeneous}. \end{aligned} \tag{3.2}$$

The combinations in (3.2) are equivalent to *sublinearity*:

$$\mathcal{F}\left(\sum_k \lambda_k X_k\right) \le \sum_k \lambda_k \mathcal{F}(X_k)$$

for $\lambda_k \ge 0$.[38]

Other important consequences of convexity emerge only in combination with closedness. One that will be applied in several ways is the following rule coming out of convex analysis.[39]

$$\begin{aligned} &\text{If } \mathcal{F} \text{ is closed convex, and if } X_0, Y, c, \text{ make the function} \\ &f(t) = \mathcal{F}(X_0 + tY) - tc \text{ be bounded above for } t \in [0, \infty), \\ &\quad \text{then } \mathcal{F}(X + tY) - tc \le \mathcal{F}(X) \text{ for all } X \text{ and } t \in [0, \infty). \end{aligned} \tag{3.3}$$

An immediate consequence, for instance, is that[40]

$$\begin{aligned} &\text{for } \mathcal{F} \text{ closed convex: if } \mathcal{F}(X) \le 0 \\ &\quad \text{whenever } X \le 0, \text{ then } \mathcal{F} \text{ is monotonic.} \end{aligned} \tag{3.4}$$

To assist with closedness, it may help to note that this property of $\mathcal{F}$ holds when $\mathcal{F}$ is continuous,[41] and moreover, as long as $\mathcal{F}$ does not take on ∞, that stronger property is automatic in broad circumstances of interest to us. Namely,[42]

$$\begin{aligned} &\mathcal{F} \text{ is continuous on } \mathcal{L}^2(\Omega) \\ &\quad \text{when } \begin{cases} \mathcal{F} \text{ is finite, convex, and closed, or} \\ \mathcal{F} \text{ is finite, convex, and monotonic, or} \\ \mathcal{F} \text{ is finite, convex, and } \Omega \text{ is finite.} \end{cases} \end{aligned} \tag{3.5}$$

Closedness can also approached through so-called "weak" convergence in place of the "strong" convergence described by (3.1), since the closedness of *convex* sets is known to be the same either way. Weak convergence of X^k to X means that $E[X^k Q] \to E[XQ]$ for all $Q \in \mathcal{L}^2$. In fact it suffices in this to restrict attention to $Q \ge 0$ with $EQ = 1$, inasmuch as linear combinations of such Q fill up all of $\mathcal{L}^2$. That will be especially meaningful in Section 6, where Q of this type will be interpreted as the density with respect to P_0 of an alternative probability measure P.

Measures of risk. The role of a measure of risk, $\mathcal{R}$, is to assign to a random variable X, standing for an uncertain "cost" or "loss", a numerical value $\mathcal{R}(X)$ that can serve as a surrogate for overall (net) cost or loss. However, the assignment must meet reasonable standards in order to make sense.

The class of *coherent* measures of risk has attracted wide attention in finance in this regard. A functional $\mathcal{R}$ belongs to this class, as introduced in [1], if it is convex and positively homogeneous (or equivalently by (3.2) subadditive and positively homogeneous), as well as monotonic, and, in addition, satisfies[43]

$$\mathcal{R}(X + C) = \mathcal{R}(X) + C \quad \text{for all } X \text{ and constants } C. \tag{3.6}$$

---

30 More explanation is provided in Section 6, which also offers motivation and examples for readers who might not be so familiar with this way of thinking.

31 The inner product between two elements X and Y of $\mathcal{L}^2(\Omega)$ is $\langle X, Y\rangle = E[XY]$.

32 It might be wondered why we insist on boundedness of second moments when requiring only $E|X| < \infty$ would cover a larger class of random variables. The main reason is that this leads to a simpler exposition in Section 6, when we come to the dualization of risk in terms of sets of probability densities Q (having $Q \ge 0$, $EQ = 1$). With the finiteness of $E|X|$ as the only requirement we would be limited there to bounded densities Q. It would be better really if we could draw on all possible densities Q, but that would force us to go to the opposite extreme of requiring X to be essentially bounded. The choice made here is a workable compromise.

33 In finite dimensions, all norms give the same convergence.

34 This feature helps to make our choice of $\mathcal{L}^2(\Omega)$ as the underlying space much less restrictive than might be imagined.

35 In expressions like this, a sum of values in $(-\infty, \infty]$ is ∞ if any of them is ∞. Also, $\lambda\infty = \infty$ for $\lambda > 0$.

36 This inequality is to be interpreted in the "almost sure" sense, meaning that the set of $\omega \in \Omega$ for which $X(\omega) \le X'(\omega)$ has probability 1.

37 This property is also called *lower semicontinuity*. A subset of $\mathcal{L}^2(\Omega)$ is closed when it contains all limits of its sequences in the sense of (3.1). For convex sets, weak limits give the same closedness as those strong limits.

38 Under the convention, if necessary, that $0\infty = 0$.

39 Apply Theorem 8.6 of [26] to the function $f(s, t) = \mathcal{F}((1-s)X_0 + sX + tY)$.

40 Consider the case of (3.3) with $X_0 = 0$, $Y \le 0$, and $c = 0$.

41 Continuity of $\mathcal{F}$ means that $\mathcal{F}(X^k) \to \mathcal{F}(X)$ whenever $X^k \to X$ as in (3.1).

42 For the first: [27, Corollary 8B]. For the second: [28, Proposition 3.1]. For the third: [26, Theorem 10.1], recalling that $\mathcal{L}^2(\Omega)$ is finite-dimensional when Ω is finite.

43 A slightly different, but ultimately equivalent property was originally formulated in [1]. Note that positive homogeneity enables the units of measurement of X to be the same as those of $\mathcal{R}(X)$.

Closedness of $\mathcal{R}$ was not mentioned in [1], but the context there supposed $\mathcal{R}$ to be finite (and actually Ω finite, too), so that closedness and even continuity of $\mathcal{R}$ were implied by coherency through (3.5).[44] Subsequent researchers considered dropping the positive homogeneity, and with it the term "coherent", speaking then of a "convex measure of risk" or a "convex risk function", cf. [10,28].[45] However, without denying the importance of these ideas, we will organize assumptions and terminology a bit differently here. The crucial role that EX has in the fundamental risk quadrangle is our guide, along with the importance of "closedness" in dealing with functionals that might take on ∞.[46]

By a *regular measure of risk* we will mean a functional $\mathcal{R}$ with values in $(-\infty, \infty]$ that is *closed convex* with

$$\mathcal{R}(C) = C \quad \text{for constants } C \tag{3.7}$$

and furthermore

$$\mathcal{R}(X) > EX \quad \text{for nonconstant } X. \tag{3.8}$$

Property (3.8) is *aversity* to risk.[47] Observe that (3.7) implies the seemingly stronger property (3.6) of [1] by the rule in (3.3)[48] and therefore entails

$$\mathcal{R}(X - EX) = \mathcal{R}(X) - EX \quad \text{for all } X \tag{3.9}$$

in particular. An advantage of stipulating (3.7) in place of (3.6) lies in motivation. The surrogate cost value that a measure of risk should assign to a random variable that always comes out with the value C ought to be C itself.

In all of the Examples 1–12 above, $\mathcal{R}$ is a regular measure of risk, and in Examples 1–3, 5–6, 10–12, $\mathcal{R}$ is also positively homogeneous. In examples 2–3, 5–10 and 12, $\mathcal{R}$ is monotonic, but in Examples 1, 4 and 11 it is not. Only the risk measures in Examples 2–3, 5–6, 10 and 12 are coherent in the sense of [1]. For $\mathcal{R} = \bar{q}_\alpha = \text{CVaR}_\alpha$ in Example 2, this was perceived from several angles that eventually came together; see [31,32,6]. For $\mathcal{R} = \bar{q}^{(p)}_\alpha$ in Example 12, the coherency was established by Krokhmal [8].

An example of a coherent measure of risk that is not regular is $\mathcal{R}(X) = EX$, which lacks aversity. On the other hand, $\mathcal{R}(X) = \text{VaR}^-_\alpha(X)$ fails to be a regular measure of risk by lacking closedness, convexity and the aversity in (3.8), in general, although it does have positive homogeneity, satisfies (3.6) and is monotonic. It fails to be a coherent measure of risk through the absence of convexity.

Measures of deviation. The role of a measure of deviation, $\mathcal{D}$, is to quantify the nonconstancy (as the uncertainty) in a random variable X.[49] By a *regular measure of deviation* we will mean a functional $\mathcal{D}$ with values in $[0, \infty]$ that is *closed convex* with

$$\begin{aligned} &\mathcal{D}(C) = 0 \quad \text{for constants } C, \\ &\text{but } \mathcal{D}(X) > 0 \text{ for nonconstant } X. \end{aligned} \tag{3.10}$$

The measures of deviation in Examples 1–12 all fit this prescription. Note that symmetry is not required: perhaps $\mathcal{D}(-X) \neq \mathcal{D}(X)$.

Measures of error. The role of a measure of error, $\mathcal{E}$, is to quantify the nonzeroness in a random variable X.[50] By a *regular measure of error* we will mean a functional $\mathcal{E}$ with values in $[0, \infty]$ that is *closed convex* with

$$\mathcal{E}(0) = 0 \quad \text{but } \mathcal{E}(X) > 0 \text{ when } X \not\equiv 0 \tag{3.11}$$

and satisfies for sequences of random variables $\{X_k\}_{k=1}^\infty$ the condition that

$$\lim_{k\to\infty} \mathcal{E}(X^k) = 0 \Longrightarrow \lim_{k\to\infty} EX^k = 0. \tag{3.12}$$

The latter requirement, meaning that random variables X with $|EX|$ bounded away from 0 cannot be arbitrarily close to 0 as measured by $\mathcal{E}(X)$, will enter into the projection from $\mathcal{E}$ to $\mathcal{D}$ that is featured on the right side of the quadrangle. It is equivalent actually to the seemingly stronger property that $\mathcal{E}(X) \geq \psi(EX)$ for a convex function ψ on $(-\infty, \infty)$ having $\psi(0) = 0$ but $\psi(t) > 0$ for $t \neq 0$.[51] In common situations it holds automatically, as for instance when Ω is finite,[52] or in the expectation case with $\mathcal{E}(X) = E[e(X)]$ for a convex function e on $(-\infty, \infty)$ having $e(0) = 0$ but $e(x) > 0$ for $x \neq 0$.[53] In Examples 1–12 every measure of error is regular, but some cases can have $\mathcal{E}(-X) \neq \mathcal{E}(X)$.

Measures of regret and relative utility. The role of a measure of regret, $\mathcal{V}$, is to quantify the displeasure associated with the mixture of potential positive, zero and negative outcomes of a random variable X that stands for an uncertain cost or loss. Regret in this sense is close to the notion of an overall penalty, but it might sometimes come out negative and therefore act as a reward. As mentioned in the introduction, regret is the flip side of relative utility. Measures of regret $\mathcal{V}$ correspond to measures of relative utility $\mathcal{U}$ through

$$\mathcal{V}(X) = -\mathcal{U}(-X), \qquad \mathcal{U}(Y) = -\mathcal{V}(-Y), \tag{3.13}$$

where Y denotes a random variable oriented toward uncertain gain instead of loss. Everything said about regret could be conveyed instead in the language of utility, but that would trigger switches of orientation between loss and gain together with tedious minus signs coming from (3.13).

By a *regular measure of regret* we will mean a functional $\mathcal{V}$ with values in $(-\infty, \infty]$ that is *closed convex*, has the aversity property that

$$\mathcal{V}(0) = 0 \quad \text{but } \mathcal{V}(X) > EX \text{ when } X \not\equiv 0, \tag{3.14}$$

and satisfies for sequences of random variables $\{X^k\}_{k=1}^\infty$ the condition that

$$\lim_{k\to\infty} [\mathcal{V}(X^k) - EX^k] = 0 \Longrightarrow \lim_{k\to\infty} EX^k = 0. \tag{3.15}$$

The limit condition parallels the one in (3.12) and likewise is automatic when Ω is finite, or in the expectation case where $\mathcal{V}(X) = E[v(X)]$ for a convex function v on $(-\infty, \infty)$ having $v(0) = 0$ but $v(x) > x$ for $x \neq 0$. All the measures of regret in Examples 1–12 are regular.

As with measures of risk $\mathcal{R}$, there is strong incentive for asking $\mathcal{V}$ also to be monotonic. That additional property holds for the measures of regret in Examples 2–3, 5–10 and 12, but not in Examples 1, 4 and 11.[54]

---

[44] Coherency was extended to general Ω in [29] with X restricted to $\mathcal{L}^\infty(\Omega)$ and $\mathcal{R}$ still finite-valued, in which case $\mathcal{R}$ is likewise continuous by Ruszczyński and Shapiro [28, Proposition 3.1]. That framework was also maintained in [10].

[45] In our view, the idea behind "coherency" is oriented to monotonicity plus convexity. In [30], risk measures satisfying the axioms of coherency except for this positive homogeneity were called *coherent in the extended sense*.

[46] Another reason is that the "convex risk measure" terminology insists on monotonicity, but we want a framework that, for the sake of broad understanding, encompasses some risk measures without monotonicity, such as $\mathcal{R}(X) = \mu(X) + \lambda\sigma(X)$.

[47] Risk measures satisfying this condition were introduced as *averse* measures of risk in [21]. A constant random variable $X \equiv C$ has $\mathcal{R}(X) = EX$ by (3.7).

[48] In (3.3) with $\mathcal{F} = \mathcal{R}$ and $X_0 = 0$, first take $Y \equiv C$ and $c = C$ for any C. Since (3.7) gives $\mathcal{R}(0 + tC) - tC = 0$, it follows that $\mathcal{R}(X + C) - C \leq \mathcal{R}(X)$ for all X and C. Applying this next to $X + C$ and $-C$ in place of X and C, get $\mathcal{R}(X) + C \leq \mathcal{R}(X + C)$, hence an equation.

[49] Deviation measures as a special class of functionals were introduced in [22] with follow-up in [21].

[50] Measures of error in such general terms were introduced in [9].

[51] From (3.12) the function $\psi(t) = \inf\{\mathcal{E}(X) \mid EX = t\}$ has these properties.

[52] The finite-dimensionality of $\mathcal{L}^2(\Omega)$ and the closed convexity $\mathcal{E}$ in combination with (3.11) ensure then that the lower level sets of $\mathcal{E}$ are compact.

[53] Then $E[e(X)] \geq e(EX)$ by Jensen's Inequality.

[54] There is potential motivation sometimes for working without such monotonicity, as will be explained in Section 5.

By a *regular measure of relative utility* we will mean a functional $\mathcal{U}$ having the "flipped" properties that correspond to those of a regular measure of regret $\mathcal{V}$ through (3.13).[55]

Quadrangle Theorem. (a) *The relations* $\mathcal{D}(X) = \mathcal{R}(X) - EX$ *and* $\mathcal{R}(X) = EX + \mathcal{D}(X)$ *give a one-to-one correspondence between regular measures of risk* $\mathcal{R}$ *and regular measures of deviation* $\mathcal{D}$. *In this correspondence,* $\mathcal{R}$ *is positively homogeneous if and only if* $\mathcal{D}$ *is positively homogeneous. On the other hand,*

$$\mathcal{R} \text{ is monotonic if and only if } \mathcal{D}(X) \le \sup X - EX \text{ for all } X. \quad (3.16)$$

(b) *The relations* $\mathcal{E}(X) = \mathcal{V}(X) - EX$ *and* $\mathcal{V}(X) = EX + \mathcal{E}(X)$ *give a one-to-one correspondence between regular measures of regret* $\mathcal{V}$ *and regular measures of error* $\mathcal{E}$. *In this correspondence,* $\mathcal{V}$ *is positively homogeneous if and only if* $\mathcal{E}$ *is positively homogeneous. On the other hand,*

$$\mathcal{V} \text{ is monotonic if and only if } \mathcal{E}(X) \le E[-X] \text{ for } X \le 0. \quad (3.17)$$

(c) *For any regular measure of regret* $\mathcal{V}$, *a regular measure of risk* $\mathcal{R}$ *is obtained by*

$$\mathcal{R}(X) = \min_C \{C + \mathcal{V}(X - C)\}. \quad (3.18)$$

If $\mathcal{V}$ *is positively homogeneous,* $\mathcal{R}$ *is positively homogeneous. If* $\mathcal{V}$ *is monotonic,* $\mathcal{R}$ *is monotonic.*

(d) *For any regular measure of error* $\mathcal{E}$, *a regular measure of deviation* $\mathcal{D}$ *is obtained by*

$$\mathcal{D}(X) = \min_C \{\mathcal{E}(X - C)\}. \quad (3.19)$$

If $\mathcal{E}$ *is positively homogeneous,* $\mathcal{D}$ *is positively homogeneous. If* $\mathcal{E}$ *satisfies the condition in* (3.17), *namely* $\mathcal{E}(X) \le E[-X]$ *for* $X \le 0$, *then* $\mathcal{D}$ *satisfies the condition in* (3.16), *namely* $\mathcal{D}(X) \le \sup X - EX$ *for all* X.

(e) *In both* (c) *and* (d), *as long as the expression being minimized is finite for some* C, *the set of* C *values for which the minimum is attained is a nonempty, closed, bounded interval.*[56] *Moreover when* $\mathcal{V}$ *and* $\mathcal{E}$ *are paired as in* (b), *the interval comes out the same and gives the associated statistic:*

$$\operatorname*{argmin}_C\{C + \mathcal{V}(X - C)\} = \mathcal{S}(X) = \operatorname*{argmin}_C\{\mathcal{E}(X - C)\},$$
$$\text{with } \mathcal{S}(X + C) = \mathcal{S}(X) + C. \quad (3.20)$$

This theorem integrates, in a new and revealing way, various results or partial results that were separately developed elsewhere, and in many instances only for positively homogeneous quantifiers. The correspondence between $\mathcal{R}$ and $\mathcal{D}$ in part (a) was officially presented in [21] after being laid out much earlier in the unpublished report [22].[57] The results in parts (d) and (e) about projecting from $\mathcal{E}$ to $\mathcal{D}$ come from [9], where they were employed in generalized linear regression.[58] The observation in part (b) immediately translates them to the results in parts (c) and (e) about projecting from $\mathcal{V}$ to $\mathcal{R}$. However, a general version of (c) in the positively homogeneous case was separately developed earlier, without that connection, by Krokhmal [8].

Although the parallel between $\mathcal{E} \to \mathcal{D}$ and $\mathcal{V} \to \mathcal{R}$, which ties the two sides of the quadrangle fully together, is mathematically elementary, it has not come into focus easily despite its conceptual significance. That, especially, is where the theorem innovates. What was absent in the past was the broad concept of a measure of regret, not limited to an expectation, and the realization it could anchor a fourth corner in the relationships, thereby serving as a conduit for bringing in "utility" beyond expected utility.

Risk measure formulas of type (3.18) with accompaniment in (3.20) have gradually emerged without any thought that they might be connected somehow with generalized regression. The first such formula was presented in [5] and its follow-up [6],[59]

$$\mathrm{CVaR}_\alpha(X) = \min_C \left\{C + \frac{1}{1-\alpha}E[X - C]_+\right\},$$
$$\mathrm{VaR}_\alpha(X) = \operatorname*{argmin}_C \left\{C + \frac{1}{1-\alpha}E[X - C]_+\right\}. \quad (3.21)$$

We later learned that the "argmin" part of this was already known in the statistics of quantile regression, cf. [12,33], but with the minimization expression differing from ours by a positive factor; the associated "min" quantity got no attention in that subject. In those days we were mainly occupied with the numerical usefulness of (3.21) in solving problems of stochastic optimization involving VaR and CVaR and were looking no further in the direction of statistics.

Earlier, on a different frontier, the concept of "optimized certainty equivalent" was defined in [34] by a trade-off formula very much like the one for getting $\mathcal{S}$ from $\mathcal{V}$ but focused on expected utility ("normalized") and maximization, instead of general regret and minimization. It was applied to problems of optimization in [35] and subsequently [36]. Much later in [7], once the theory of risk measures had come into development, the "min" quantity in the trade-off received attention alongside of the "argmin", and (3.21) could be cast as a special case of their previous work with expected utility. An important feature of that work, brought out further in [7], was duality with notions of information and entropy.[60]

In [8] a much wider class of trade-off formulas for risk measures was studied with the aim of generalizing (3.21) through $\mathcal{V}$-type expressions not restricted to the expectation case. In that research, as in [7], no connections with statistical theory were contemplated. In other words, the bottom line of the quadrangle was still out of sight.

It is convenient to speak of the quantifiers at the corners of the fundamental quadrangle, under the relations in Diagram 3, as constituting a *quadrangle quartet* $(\mathcal{R}, \mathcal{D}, \mathcal{V}, \mathcal{E})$ with statistic $\mathcal{S}$. In the regular case portrayed in the Quadrangle Theorem, it is a *regular* quadrangle quartet. The most attractive case adds monotonicity to $\mathcal{R}$ and $\mathcal{V}$ along with the corresponding

---

55 More details on this will be provided in Section 4.

56 Typically this interval reduces to a single point.

57 As in those works, even though they only looked at the positively homogeneous case, the justification of (3.17) follows by applying (3.4) to $\mathcal{F} = \mathcal{R}$. The justification of (3.13) works the same way with $\mathcal{F} = \mathcal{V}$ in (3.4).

58 The only real effort in the proof of the projection claims is in showing that, when $\mathcal{D}$ comes from (3.19), the minimum over C is attained and $\mathcal{D}$ inherits the closedness of $\mathcal{E}$. This draws on (3.12). The argument in [9] utilized positive homogeneity, but it is readily generalized as follows through the existence under (3.12) of a convex function ψ with $\psi(0) = 0$, $\psi(t) > 0$ for $t \ne 0$, such that $\mathcal{E}(X) \ge \psi(EX)$. The level sets $\{t \mid \psi(t) \le c\}$ are then bounded.

Observe first that if a sequence of finite error values $\mathcal{E}(X - C^k)$ approaches the minimum with respect to C, it is a bounded sequence and therefore, since $\mathcal{E}(X - C^k) \ge \psi(EX - C^k)$, the sequence of expected values $E[X - C^k]$ is bounded. Then the sequence $\{C^k\}_{k=1}^\infty$ is bounded, so a subsequence will converge to some C. That C gives the minimum, due to the closedness of $\mathcal{E}$.

Next fix a value $c \in \mathbb{R}$ and suppose that $X^k \to X$ with $\mathcal{D}(X^k) \le c$ for $k = 1, 2, \ldots$. The issue is whether $\mathcal{D}(X) \le c$. For each k there is a C^k with $\mathcal{D}(X^k) = \mathcal{E}(X^k - C^k)$, and those error values are bounded then by c. In consequence, the sequence of values $E[X^k - C^k]$ is bounded. Since $X^k \to X$, hence $EX^k \to EX$, it follows that a subsequence of $\{C^k\}_{k=1}^\infty$ has to converge to some C, in which case the corresponding subsequence of $\{X^k - C^k\}_{k=1}^\infty$ converges to $X - C$. The closedness of $\mathcal{E}$ ensures that $\mathcal{E}(X - C) \le c$ and hence $\mathcal{D}(X) \le c$, as required.

59 In some papers in this area the random variables X were taken as representing uncertain "gains" instead of "losses". The resulting formulas are of course equivalent in that case, but minus signs have to be juggled in the translation.

60 Here, see the end of Section 6.

properties of $\mathcal{D}$ and $\mathcal{E}$ in (3.16) and (3.17); we will then call the quartet *monotonic*. On the other hand, in the case where the four quantifiers are positively homogeneous we will speak of a quartet with positive homogeneity.

Although good examples of regular quadrangle quartets with and without monotonicity have been provided in Section 2, the question arises of how additional examples might be constructed. We round out this section with three results which can assist in that direction.

The first one is elementary but puts into the proper perspective of an entire quadrangle the operation of blending risk with expectation that is seen in the formula it gives for $\mathcal{R}(X)$. Such blending, for instance with $\mathcal{R}_0(X) = \mathrm{CVaR}_\alpha(X)$, has gained some attention in finance.

Scaling Theorem. *Let $(\mathcal{R}_0, \mathcal{D}_0, \mathcal{V}_0, \mathcal{E}_0)$ be a regular quadrangle quartet with statistic $\mathcal{S}_0$ and consider any $\lambda \in (0, \infty)$. Then a regular quadrangle quartet $(\mathcal{R}, \mathcal{D}, \mathcal{V}, \mathcal{E})$ with statistic $\mathcal{S}$ is given by*

$$\begin{aligned} &\mathcal{S}(X) = \mathcal{S}_0(X),\\ &\mathcal{R}(X) = (1-\lambda)EX + \lambda\mathcal{R}_0(X), \qquad \mathcal{D}(X) = \lambda\mathcal{D}_0(X),\\ &\mathcal{V}(X) = (1-\lambda)EX + \lambda\mathcal{V}_0(X), \qquad \mathcal{E}(X) = \lambda\mathcal{E}_0(X). \end{aligned} \tag{3.22}$$

Another is given by

$$\begin{aligned} &\mathcal{S}(X) = \lambda\mathcal{S}_0(\lambda^{-1}X),\\ &\mathcal{R}(X) = \lambda\mathcal{R}_0(\lambda^{-1}X), \qquad \mathcal{D}(X) = \lambda\mathcal{D}_0(\lambda^{-1}X),\\ &\mathcal{V}(X) = \lambda\mathcal{V}_0(\lambda^{-1}X), \qquad \mathcal{E}(X) = \lambda\mathcal{E}_0(\lambda^{-1}X). \end{aligned} \tag{3.23}$$

Monotonicity and positive homogeneity are preserved in these constructions, except that monotonicity requires $\lambda \geq 1$ in (3.22).

Scaling as in (3.22) is present in Examples 1 and 1′, and could very well be added to Examples 2 and 3. The alternative form in (3.23) provides an enrichment to Examples 8 and 9.

Mixing Theorem. *For $k = 1, \ldots, r$ let $(\mathcal{R}_k, \mathcal{D}_k, \mathcal{V}_k, \mathcal{E}_k)$ be a regular quadrangle quartet with statistic $\mathcal{S}_k$, and consider any weights $\lambda_k > 0$ with $\lambda_1 + \cdots + \lambda_r = 1$. A regular quadrangle quartet $(\mathcal{R}, \mathcal{D}, \mathcal{V}, \mathcal{E})$ with statistic $\mathcal{S}$ is given then by*

$$\begin{aligned} &\mathcal{S}(X) = \lambda_1\mathcal{S}_1(X) + \cdots + \lambda_r\mathcal{S}_r(X),\\ &\mathcal{R}(X) = \lambda_1\mathcal{R}_1(X) + \cdots + \lambda_r\mathcal{R}_r(X),\\ &\mathcal{D}(X) = \lambda_1\mathcal{D}_1(X) + \cdots + \lambda_r\mathcal{D}_r(X),\\ &\mathcal{V}(X) = \min_{B_1,\ldots,B_r}\left\{\sum_{k=1}^r \lambda_k\mathcal{V}_k(X - B_k) \mid \sum_{k=1}^r \lambda_k B_k = 0\right\},\\ &\mathcal{E}(X) = \min_{B_1,\ldots,B_r}\left\{\sum_{k=1}^r \lambda_k\mathcal{E}_k(X - B_k) \mid \sum_{k=1}^r \lambda_k B_k = 0\right\}. \end{aligned} \tag{3.24}$$

Moreover $(\mathcal{R}, \mathcal{D}, \mathcal{V}, \mathcal{E})$ is monotonic if every $(\mathcal{R}_k, \mathcal{D}_k, \mathcal{V}_k, \mathcal{E}_k)$ is monotonic, and $(\mathcal{R}, \mathcal{D}, \mathcal{V}, \mathcal{E})$ is positively homogeneous if every $(\mathcal{R}_k, \mathcal{D}_k, \mathcal{V}_k, \mathcal{E}_k)$ is positively homogeneous.

This generalizes a result in [9] which dealt only with positively homogeneous quantifiers.[61] The quadrangle in Example 10 illustrates it for a particular case.

[61] The proof is essentially the same as in that case, the main task being to demonstrate that $\mathcal{R}$ and $\mathcal{D}$ are closed and the minimum over $B_1, \ldots, B_r$ is attained. The argument follows the pattern we have indicated above for the projection part of the Quadrangle Theorem, making use of bounds $\mathcal{E}_k(X) \geq \psi_k(EX)$ coming from the lines after (3.12).

Reverting Theorem. *For $i = 1, 2$, let $(\mathcal{R}_i, \mathcal{D}_i, \mathcal{V}_i, \mathcal{E}_i)$ be a regular quadrangle quartet with statistic $\mathcal{S}_i$. Then a regular quadrangle quartet $(\mathcal{R}, \mathcal{D}, \mathcal{V}, \mathcal{E})$ with statistic $\mathcal{S}$ is given by*

$$\begin{aligned} &\mathcal{S}(X) = \frac{1}{2}[\mathcal{S}_1(X) - \mathcal{S}_2(-X)],\\ &\mathcal{R}(X) = EX + \frac{1}{2}[\mathcal{R}_1(X) + \mathcal{R}_2(-X)],\\ &\mathcal{D}(X) = \frac{1}{2}[\mathcal{D}_1(X) + \mathcal{D}_2(-X)] = \frac{1}{2}[\mathcal{R}_1(X) + \mathcal{R}_2(-X)],\\ &\mathcal{V}(X) = EX + \min_B\left\{\frac{1}{2}[\mathcal{V}_1(B+X) + \mathcal{V}_2(B-X)] - B\right\},\\ &\mathcal{E}(X) = \min_B\left\{\frac{1}{2}[\mathcal{E}_1(B+X) + \mathcal{E}_2(B-X)]\right\}. \end{aligned} \tag{3.25}$$

Positive homogeneity is preserved in this construction, but not monotonicity.

Example 11 illustrates a case where $(\mathcal{R}_1, \mathcal{D}_1, \mathcal{V}_1, \mathcal{E}_1)$ and $(\mathcal{R}_2, \mathcal{D}_2, \mathcal{V}_2, \mathcal{E}_2)$ coincide. The proof of the Reverting Theorem takes advantage of bounds $\mathcal{E}_i(X) \geq \psi_i(EX)$ produced from (3.12).[62]

A further operation that can be performed on risk measures is "inf-convolution", cf. [37]. This could likewise be articulated in a theorem along these lines.

4. Further model-promoting results and interpretations

The general facts in Section 3 will be supplemented in this section by more detail in the expectation case. Claims made about the examples of expectation quadrangles in Section 2 will in that way be confirmed. Insight will be provided also into the pattern of regret versus utility, even outside the expectation case, and how it can affect the $\mathcal{D}$–$\mathcal{E}$ side of the quadrangle.

In relying on (3.13) for a one-to-one correspondence between regular measures of regret $\mathcal{V}$ and regular measures of relative utility $\mathcal{U}$, we are in particular replacing the convexity of $\mathcal{V}$ with the concavity of $\mathcal{U}$ and requiring, for a random variable Y oriented toward gain, that

$$\mathcal{U}(0) = 0 \quad \text{but } \mathcal{U}(Y) < EY \text{ when } Y \not\equiv 0. \tag{4.1}$$

This is where the term "relative" comes in. The gain in Y needs to be viewed as *gain relative to some benchmark*. That contrasts with the way utility theory is ordinarily articulated in terms of the "absolute" utility of an outcome. But practitioners appreciate nowadays that investors, for instance, are highly influenced by benchmarks in their attitudes toward gain or loss.

The case of expected utility, focused on $E[u(Y)]$ for a one-dimensional utility function u giving $u(y)$ for a sure gain y, serves well in explaining this. A large body of traditional theory in finance, laid out authoritatively in [10], looks toward maximizing such an expression under various side conditions in putting together a good portfolio. The utility function u captures the preferences of an investor, and the expectation deals with the uncertainty when the gain y turns into a random variable Y. Standard functions u have logarithmic forms and the like, and there is often nothing "relative" about them.

In order to have a functional $\mathcal{U}(Y) = E[u(Y)]$ satisfy (4.1) and be closed concave,[63] the natural specialization is to require u to be

[62] It starts with a direct calculation of the minimum of $\mathcal{E}(X - C)$ over C with the $\min_B$ expression for $\mathcal{E}$ inserted. A change of variables $C_1 = C - B$, $C_2 = -C - B$, shows that this yields the claimed $\mathcal{S}$, and $\mathcal{D}$. The corresponding $\mathcal{R}$ and $\mathcal{V}$ are confirmed then from the quadrangle formulas.

[63] Closed concavity requires the "upper" level sets of type $\geq c$ to be closed for all $c \in \mathbb{R}$, in contrast to closed convexity, which requires all "lower" level sets of type $\leq c$ to be closed.

a function of y with

$$u \text{ closed concave and } u(0) = 0 \quad \text{but } u(y) < y \text{ when } y \neq 0. \tag{4.2}$$

Again, the sense in that would come from a benchmark interpretation, namely that y no longer stands for an amount of money received in the future but rather an increment (positive or negative) to some reference amount. A utility function satisfying (4.2), but with "$<$" weakened to "$\leq$", is a *normalized* utility in the terminology of [7]. Normalization to create these properties is always possible in the expectation case because, in theory, as far as generating a preference ordering for y values is concerned, a utility u is only determined up to translations and an arbitrary scaling factor.[64] For our quadrangle scheme, however, such normalization is not merely a convenience but essential. Expected utility depends not only on the ordering induced by u on $(-\infty, \infty)$, but also on the "curvature" aspects of u, and the choice of a benchmark can have a large impact on that, apart from some special cases.

A utility function u satisfying (4.2) is paired with a regret function v satisfying

$$v \text{ closed convex and } v(0) = 0 \quad \text{but } v(x) > x \text{ when } x \neq 0 \tag{4.3}$$

under the correspondence[65]

$$v(x) = -u(-x), \qquad u(y) = -v(-y). \tag{4.4}$$

The properties in (4.3) are needed for $\mathcal{V}(X) = E[v(X)]$ to be a *regular* measure of regret. They are crucial moreover in the correspondence between $\mathcal{V}$ and $\mathcal{E}$ at the bottom of the quadrangle in making $\mathcal{E}(X) = E[e(X)]$ be a regular measure of error paired with $\mathcal{V}(X) = E[v(X)]$ under the relations

$$e(x) = v(x) - x, \qquad v(x) = x + e(x), \tag{4.5}$$

which entail having

$$e \text{ closed convex and } e(0) = 0 \quad \text{but } e(x) > 0 \text{ when } x \neq 0. \tag{4.6}$$

The condition on the utility function u in (4.2) implies that $u'(0) = 1$ when u is differentiable at 0, but it is important to realize that u might not be differentiable at 0, and this could even be desirable. From concavity, u is sure at least to have right derivatives $u'_+(y)$ and left derivatives $u'_-(y)$ satisfying $u'_-(y) \geq u'_+(y)$, usually with equality, but still maybe with $u'_-(0) > u'_+(0)$. This would mean that, in terms of relative utility, *the pain of a marginal loss relative to the benchmark is greater than pleasure of a marginal gain relative to the benchmark*. Just such a disparity in reactions to gains and losses is seen in practice and reflects, at least in part, the observations in [38].

In translating this from a concave utility function u to a convex regret function v as in (4.3), we have, of course, right derivatives $v'_+(x)$ and $v'_-(x)$ satisfying $v'_-(x) \leq v'_+(x)$, usually with equality, but perhaps with $v'_-(0) < v'_+(0)$. However, something more needs to be understood in connection with the ability of v to take on ∞ and how that affects the way derivatives are treated in the formulas of the theorem below.

The convexity of v implies that the effective domain $\operatorname{dom} v = \{x \mid v(x) < \infty\}$ is an interval in $(-\infty, \infty)$ (not necessarily closed or bounded). If x is the right endpoint of $\operatorname{dom} v$, the definition of the right derivative naturally gives $v'_+(x) = \infty$; but just in case of doubt in some formula, this is also the interpretation to give of $v'_+(x)$ when x is off to the right of $\operatorname{dom} v$.[66] Likewise, if x is the left endpoint of $\operatorname{dom} v$, or further to the left, then $v'_-(x) = -\infty$.

These are the patterns also for an error function e as in (4.6).

For the fundamental quadrangle of risk, the consequences of these facts in the expectation case are summarized as follows.

Expectation Theorem. *For functions v and e on $(-\infty, \infty)$ related by (4.5), the properties in (4.3) amount to those in (4.6) and ensure that the functionals*

$$\mathcal{V}(X) = E[v(X)], \qquad \mathcal{E}(X) = E[e(X)], \tag{4.7}$$

form a corresponding pair consisting of a regular measure of regret and a regular measure of error.[67] *For $X \in V = \operatorname{dom} \mathcal{V} = \operatorname{dom} \mathcal{E}$ let $C^+(X) = \sup\{C \mid X - C \in V\}$ and $C^-(X) = \inf\{C \mid X - C \in V\}$. The associated statistic $\mathcal{S}$ in the quadrangle generated from $\mathcal{V}$ and $\mathcal{E}$ is characterized then by*

$$\begin{aligned} \mathcal{S}(X) &= \big\{C \mid E[e'_-(X - C)] \leq 0 \leq E[e'_+(X - C)]\big\} \\ &= \big\{C \mid E[v'_-(X - C)] \leq 1 \leq E[v'_+(X - C)]\big\} \end{aligned} \tag{4.8}$$

subject to the modification that, in both cases, the right side is replaced by ∞ if $C \leq C^-(X)$ and the left side is replaced by $-\infty$ if $C \geq C^+(X)$. The quadrangle is completed then by setting

$$\begin{aligned} &\mathcal{D}(X) = E[e(X - C)] \quad \textit{and} \\ &\mathcal{R}(X) = C + E[v(X - C)] \quad \textit{for any/all } C \in \mathcal{S}(X). \end{aligned} \tag{4.9}$$

Having $\mathcal{V}$ and $\mathcal{R}$ be monotonic corresponds (in tandem with convexity) to having $v(x) \leq 0$ when $x < 0$, or equivalently $e(x) \leq -x$ when $x < 0$. Positive homogeneity holds in the quadrangle if and only if v and e have graphs composed of two linear pieces kinked at 0.

Beyond the aspects of this theorem that are already evident, the key ingredient is establishing (4.8). This is carried out by calculating that the right and left derivatives of the convex function $\phi(C) = E[e(X - C)]$ from their definitions and noting that C belongs to $\operatorname{argmin} \phi$ if and only if $\phi'_-(C) \leq 0 \leq \phi'_+(C)$. In situations where v and e are differentiable, the double inequalities in (4.8) can be replaced simply by the equations $E[e'(X - C)] = 0$ and $E[v'(X - C)] = 1$.

We proceed now to illustrate the Expectation Theorem by applying it to justify the details of the examples in Section 2 that belong to the expectation case.

Quantile-based quadrangle, Example 2 (*including* Example 3):

$$e(x) = \frac{\alpha}{1-\alpha} \max\{0, x\} + \max\{0, -x\},$$

$$v(x) = \frac{1}{1-\alpha} \max\{0, x\}, \qquad u(y) = \frac{1}{1-\alpha} \min\{0, y\}.$$

We have $V = \mathcal{L}^2(\Omega)$, $C^+(X) = \infty$, $C^-(X) = -\infty$, and

$$v'_+(x) = \begin{cases} \dfrac{1}{1-\alpha} & \text{if } x \geq 0, \\ 0 & \text{if } x < 0, \end{cases} \qquad v'_-(x) = \begin{cases} \dfrac{1}{1-\alpha} & \text{if } x > 0, \\ 0 & \text{if } x \leq 0, \end{cases}$$

with a gap between left and right derivatives occurring only at $x = 0$. Then with $F_X^-(C)$ denoting the left limit of F_X at C (the right limit $F_X^+(C)$ being just $F_X(C)$), we get

$$E[v'_-(X - C)] = \frac{1}{1-\alpha} \operatorname{prob}\{X > C\} = 1 - F_X(C),$$

$$E[v'_+(X - C)] = \frac{1}{1-\alpha} \operatorname{prob}\{X \geq C\} = 1 - F_X^-(C).$$

It follows thereby from (4.8) that $\mathcal{S}(X) = \{C \mid F_X^-(C) \leq \alpha \leq F_X(C)\}$ and therefore $\mathcal{S}(X) = q_\alpha(X)$. Applying (4.10) yields $\mathcal{R}(X) = C + \frac{1}{1-\alpha} E \max\{0, X - C\} = C + \frac{1}{1-\alpha} \int_{(C,\infty)} (x - C) dF_X(x)$. Since the probability of (C, ∞) is $1 - F_X(C)$, this equals $\frac{1}{1-\alpha}[(F_X(C) - \alpha)]C + \int_{[C,\infty)} x dF_X(x)$, which is the expectation of X with respect to its

[64] Outside of the expectation case, it is still possible to shift to $\mathcal{U}(0) = 0$ as a "normalization", but rescaling is insufficient to get to $\mathcal{U}(Y) \leq EY$.

[65] In this correspondence the graphs of v and u reflect to each other through the origin of $\mathbb{R}^2$.

[66] The issue is that a random X might produce such an outcome with probability 0, and yet one still needs to know how to think of the formula.

[67] Also, $\mathcal{V}$ corresponds then to a regular measure of relative utility $\mathcal{U}$ given by $\mathcal{U}(Y) = E[u(Y)]$ under (4.4) via (4.2).

"α-tail distribution" as defined in [6] and used there to properly define $\bar{q}_\alpha(X)$ even under the possibility that $F_X(C) > \alpha$.

Worst-case-based quadrangle, Example 5:

$$e(x) = \begin{cases} -x & \text{if } x \le 0 \\ \infty & \text{if } x > 0, \end{cases} \qquad v(x) = \begin{cases} 0 & \text{if } x \le 0 \\ \infty & \text{if } x > 0, \end{cases}$$

$$u(y) = \begin{cases} -\infty & \text{if } y < 0 \\ 0 & \text{if } y \ge 0. \end{cases}$$

We have $V = \mathcal{L}^2_-(\Omega)$, $C^+(X) = \infty$, $C^-(X) = \sup X$. In the v part of (4.8) the left side equals 0 always and the right side equals 0 if $C < \sup X$ but (through the prescribed modification) equals ∞ if $C = \sup X$. Therefore, $C = \sup X$ is the unique element of $\mathcal{S}(X)$ (when that is finite).

Truncated-mean-based quadrangle, Example 7:

$$e(x) = \begin{cases} |x| - \dfrac{\beta}{2} & \text{if } |x| \ge \beta, \\ \dfrac{1}{2\beta}x^2 & \text{if } |x| \le \beta, \end{cases}$$

$$v(x) = \begin{cases} -\dfrac{\beta}{2} & \text{if } x \le -\beta, \\ x + \dfrac{1}{2\beta}x^2 & \text{if } |x| \le \beta, \\ 2x - \dfrac{\beta}{2} & \text{if } x \ge \beta, \end{cases}$$

$$u(y) = \begin{cases} 2y + \dfrac{\beta}{2} & \text{if } y \le -\beta, \\ y - \dfrac{1}{2\beta}y^2 & \text{if } |y| \le \beta, \\ \dfrac{\beta}{2} & \text{if } y \ge \beta. \end{cases}$$

This time, $V = \mathcal{L}^2(X)$, so $C^+(X) = \infty$ and $C^-(X) = -\infty$. The statistic is determined by solving $E[e'(X - C)] = 0$ for C, and this gives the result described because

$$\beta e'(x) = T_\beta(x) = \begin{cases} \beta & \text{if } x \ge \beta, \\ x & \text{if } -\beta \le x \le \beta, \\ -\beta & \text{if } x \le -\beta. \end{cases}$$

Log-exponential-based quadrangle, Example 8:

$$e(x) = \exp x - x - 1, \qquad v(x) = \exp x - 1,$$
$$u(y) = 1 - \exp(-y).$$

Here $V = \{X \mid E[\exp X] < \infty\}$. Because $E[\exp(X - C)] = \exp(-C)E[\exp X]$, we have $C^+(X) = \infty$ and $C^-(X) = -\infty$ for any $X \in V$, so the need for a modification of the bounds in (4.8) is avoided. Indeed, since $v'(x) = \exp x$, we just have an equation to solve for C, namely $E[\exp(X - C)] = 1$. This equation can be rewritten as $E[\exp X] = \exp C$, which yields $C = \log E[\exp X]$ as $\mathcal{S}(X)$. Substituting that into $C + \mathcal{V}(X - C)$, we get $\mathcal{R}(X) = \log E[\exp X]$ and the quadrangle is confirmed.

Rate-based quadrangle, Example 9:

$$e(x) = \begin{cases} \log \dfrac{1}{1-x} - x & \text{if } x < 1, \\ \infty & \text{if } x \ge 1, \end{cases}$$

$$v(x) = \begin{cases} \log \dfrac{1}{1-x} & \text{if } x < 1, \\ \infty & \text{if } x \ge 1, \end{cases}$$

$$u(y) = \begin{cases} \log(1+y) & \text{if } y > -1, \\ -\infty & \text{if } y \le -1. \end{cases}$$

Here $V = \{X < 1 \mid E[\log \frac{1}{1-X}] < \infty\}$, so $C^+(X) = 1 - \sup X$ and $C^-(X) = -\infty$. Because v is differentiable (where finite), we have an equation to solve in (4.8): $E[\frac{1}{1-(X-C)}] = 1$. The solution is the statistic $\mathcal{S}(X)$.

Quadrangles from kinked utility and regret. More examples beyond the differentiable case of the Expectation Theorem can be produced by starting from an "absolute" utility function $u_0(y_0)$ that is differentiable, increasing and strictly concave, introducing a benchmark value B, and a "kink" parameter $\delta > 0$, and defining

$$u(y) = \frac{u_0(y + B) - u_0(B)}{u_0'(B)} + \delta \min\{0, y\}. \tag{4.10}$$

This will satisfy $u(0) = 0$ and $u(y) < y$ when $y \neq 0$, and it will be differentiable when $y \neq 0$, but have

$$u'_+(0) = 1 \text{ but } u'_-(0) = 1 + \delta. \tag{4.11}$$

The kink parameter δ models the extra pain experienced in falling short of the benchmark, in contrast to the milder pleasure experienced in exceeding it. From this u it is straightforward to pass to the corresponding v, e, and the full quadrangle associated with them by the theorem. In general, that quadrangle will depend on both B and δ, but in special situations like CARA or HARA utilities[68] the B dependence can drop out or reduce to simple rescaling.

The surprising fact is that all such manipulations are propagated by the quadrangle scheme into applications not just to risk management and optimization but also to statistical estimation. Those applications will be discussed further in Section 5.

General interpretations of the quadrangle "statistic". Returning finally to the general level of the correspondence $\mathcal{U} \leftrightarrow \mathcal{V}$ between relative utility and regret in (3.13) we look at ways of interpreting the trade-off formula $\mathcal{R}(X) = \min_C\{C + \mathcal{V}(X - C)\}$. Through a change of variables $Y = -X$, $W = -C$, switching loss to gain, this corresponds to

$$-\mathcal{R}(-Y) = \max_W\{W + \mathcal{U}(Y - W)\}. \tag{4.12}$$

Considerations were focused in [7] on the expectation case, but an interpretation suggested there works well for (4.12) in general. To begin with, note that in adding W to $\mathcal{U}(Y - W)$ it is essential that W be measured in the same units as $\mathcal{U}(Y - W)$, and moreover they have to be the same units as those of Y. A simple case where this makes perfect sense is the one in which the units are money units, like dollars. Then W represents an income that is certain, whereas $Y - W$ is residual income that is uncertain; $\mathcal{U}$ assigns to that uncertain income an equally desirable amount of certain income in something akin to a discount. This leads, in [7], a W giving the max in (4.12) being called an *optimized certainty equivalent* for Y.

Much the same can be said about the regret version of trade-off, $\mathcal{R}(X) = \min_C\{C + \mathcal{V}(X - C)\}$. There, C is a loss that is certain, $X - C$ is a residual loss that is uncertain. The regret measure $\mathcal{V}$ assigns to $X - C$ an amount of money that could be deemed adequate as immediate compensation for taking on the burden of $X - C$. It is possible to elaborate this with ideas of insurance, insurance premium, "deductibles", and so forth. For some insurance interpretations in the utility context of (4.12), see [7].

Although these "min" formulas and interpretations are natural in their own right, the special insight from the risk quadrangle, namely that they have a parallel life in theoretical statistics, is new.

[68] See [10, pp. 68–69].

5. Quadrangle roles in optimization and regression

Applications involving the quantifiers on both sides of the risk quadrangle have provided key motivation and guidance for the theory that has been laid out. The purpose of this section is to explain that background and indicate advances that the theory now brings.

Optimization. Risk in the sense quantified by a risk measure $\mathcal{R}$ is central in the management and control of cost or loss. For a hazard variable X, the crucial issue there is how to model a "soft" upper bound, i.e., a condition that the outcomes of X be "adequately" $\leq C$ for some C. As already explained in Section 1, the broad prescription for handling this is to pass to a numerical inequality $\mathcal{R}(X) \leq C$ through some choice of a risk measure $\mathcal{R}$, and many possibilities for $\mathcal{R}$ have been offered. Of course C can be taken to be 0 without any real loss of generality.

A choice of $\mathcal{R}$ corresponds to an expression of preferences toward risk, but it might not yet be clear why some measures of risk are better motivated or computationally more tractable than others. The key challenge is that most applications require more than just looking at $\mathcal{R}(X)$ for a single X, as far as optimization is concerned. Usually instead, there is a random variable that depends on parameters $x_1, \ldots, x_n$. We have $X(x_1, \ldots, x_n)$ and it becomes important to know how the numerical surrogate $\mathcal{R}(X(x_1, \ldots, x_n))$ depends on $x_1, \ldots, x_n$. This is where favorable conditions imposed on $\mathcal{R}$, like convexity and monotonicity, are indispensable.

Motivations in optimization modeling are important in particular. For insight, consider first a standard type of *deterministic* optimization problem, *without uncertainty*, in which $x = (x_1, \ldots, x_n)$ is the decision vector, namely

$$\begin{aligned} &\text{minimize } f_0(x) \text{ over all } x \in S \subset \mathbb{R}^n \\ &\quad \text{subject to } f_i(x) \leq 0 \quad \text{for } i = 1, \ldots, m. \end{aligned} \tag{$\mathcal{P}$}$$

A decision x selected from the set S results in *numerical values* $f_0(x), f_1(x), \ldots, f_m(x)$, which can be subjected to the usual techniques of optimization methodology. Suppose next, though, that these cost-like expressions are *uncertain* through dependence on additional variables – random variables – whose realizations will not be known until later. A decision x merely results then in *random variables*[69]

$$X_0(x) = \underline{f}_0(x), \qquad X_1(x) = \underline{f}_1(x), \ldots, X_m(x) = \underline{f}_m(x), \tag{5.1}$$

which can only be *shaped in their distributions* through the choice of x, not pinned down to specific values. Now there is no longer a single, evident answer to how optimization should be viewed, but risk measures can come to the rescue.

As proposed in [30], one can systematically pass to a *stochastic* optimization problem in the format[70]

$$\begin{aligned} &\text{minimize } \overline{f}_0(x) = \mathcal{R}_0(\underline{f}_0(x)) \text{ over } x \in S \\ &\quad \text{subject to } \overline{f}_i(x) = \mathcal{R}_i(\underline{f}_i(x)) \leq 0, \quad i = 1, \ldots, m, \end{aligned} \tag{$\underline{\mathcal{P}}$}$$

in which an individually selected "measure of risk" $\mathcal{R}_i$ has been combined with each $\underline{f}_i(x)$ to arrive at a numerical (nonrandom) function $\overline{f}_i$ of the decision vector x.[71]

An issue that must then be addressed is how the properties of $\overline{f}_i(x)$ with respect to x relate to those of $\underline{f}_i(x)$ through the choice of $\mathcal{R}_i$, and whether those properties are conducive to good optimization modeling and solvability. This is not to be taken for granted, because seemingly attractive examples like $\mathcal{R}_i(X) = EX + \lambda_i \sigma(X)$ with $\lambda_i > 0$ or $\mathcal{R}_i(X) = q_{\alpha_i}(X) = \mathrm{VaR}_{\alpha_i}(X)$ with $0 < \alpha_i < 1$ are known to suffer from troubles with "coherency" in the sense of [1].

Convexity Theorem.[72] *In problem ($\underline{\mathcal{P}}$), the convexity of $\overline{f}_i(x)$ with respect to x is assured if $\underline{f}_i(x)$ is linear in x and $\mathcal{R}_i$ is a regular measure of risk, or if $\underline{f}_i(x)$ is convex in x and $\mathcal{R}_i$ is, in addition, a monotonic measure of risk.*[73]

The huge advantage of having the functions $\overline{f}_i$ be convex is that then, with the set S also convex, ($\underline{\mathcal{P}}$) is an *optimization problem of convex type*. Such problems are vastly easier to solve in computation.

The use of $\mathcal{R}_i(X) = q_{\alpha_i}(X) = \mathrm{VaR}_{\alpha_i}(X)$ in this setting could destroy whatever underlying convexity with respect to $x = (x_1, \ldots, x_n)$ might be available in the problem data, because this measure of risk lacks convexity; it is not regular and not coherent. The shortcoming of $\mathcal{R}_i(X) = EX + \lambda_i \sigma(X)$ is different: it fails in general to be monotonic. The absence of monotonicity threatens the transmittal of convexity of $\underline{f}_i(x)$ to $\overline{f}_i(x)$. However, $\overline{f}_i(x)$ can still be convex in x, on the basis of the Convexity Theorem, as long as $\underline{f}_i(x)$ is linear in x. This could be useful in applications to financial optimization, because linearity with respect to x, as a vector of "portfolio weights", is often encountered there.

Another example of a measure of risk that is regular without being monotonic is the reverted CVaR in Example 11: $\mathcal{R}_i(X) = EX + \frac{\lambda}{2}[\mathrm{CVaR}_{\alpha_i}(X) + \mathrm{CVaR}_{\alpha_i}(-X)]$. Once more, although this choice would not preserve convexity in general, it would do so when $\underline{f}_i(x)$ is linear in x.

A question of modeling motivation must be confronted here. Why would one ever wish to use in a stochastic optimization problem ($\underline{\mathcal{P}}$) a regular risk measure that is not monotonic, even in applications with linearity in x, when so many choices do have that property? An interesting justification can actually be given, which could sometimes make sense in finance, at least. The rationale has to do with skepticism about the data in the model and especially a wish to not rely too much on data in the extreme lower tail of a cost distribution. Optimization with today's data will be succeeded by optimization with tomorrow's data, all data being imperfect. It would be wrong to swing very far in response to ephemeral changes, at least in formulating the objective function $\overline{f}_0(x) = \mathcal{R}_0(\underline{f}_0(x))$.

The following idea comes up: replace this objective, in the case of a regular monotonic measure of risk $\mathcal{R}_0$, by a measure of risk

[69] We employ underbars in this discussion to indicate uncertainty. The overbars appearing later emphasize that the random variable depending on x has been converted to a nonrandom numerical function of x.

[70] If taken too literally, this prescription could be simplistic. When uncertainty is present, much closer attention must be paid to whether the objective and constraint structure in the deterministic formulation itself was well chosen. The effects of possible recourse actions when constraints are violated may need to be brought in. Whether risk measures should be applied to the f_i's individually or to a combination passed through some joint expression must be considered as well.

[71] The constraint modeling in ($\underline{\mathcal{P}}$) follows the prescription that $\mathcal{R}_i(\underline{f}_i(x)) \leq 0$ provides a rigorous interpretation to the desire of having $\underline{f}_i(x)$ "adequately" ≤ 0, but the motivation for the treatment of the objective in ($\underline{\mathcal{P}}$) may be less clear. Actually, it follows the same prescription. Choosing x to minimize $\mathcal{R}_0(\underline{f}_0(x))$ can be identified with choosing a pair (x, C_0) subject to $\mathcal{R}_0(\underline{f}_0(x)) \leq C_0$ so as to get C_0 as low as possible, and the inequality $\mathcal{R}_0(\underline{f}_0(x)) \leq C_0$ models having $\underline{f}_0(x)$ "adequately" $\leq C_0$. This is valuable in handling the dangers of "cost overruns".

[72] This extends, in an elementary way, a principle in [30].

[73] Convexity of the random variable $\underline{f}_i(x)$ with respect to x refers to having $\underline{f}_i((1-\lambda)x_0 + \lambda x_1) \leq (1-\lambda)\underline{f}_i(x_0) + \lambda \underline{f}_i(x_1)$ as a relation among random variables, i.e., with "almost surely" coming in.

having the form

$$\tilde{\mathcal{R}}_0(X) = \mathcal{R}_0(X) + \mathcal{D}(X)$$
for some regular measure of deviation $\mathcal{D}$. (5.2)

This would be another regular measure of risk, even if not monotonic. The deviation term would be designed to have a "stabilizing" effect.

If a choice like $\mathcal{R}_i(X) = q_{\alpha_i}(X) = \mathrm{VaR}_{\alpha_i}(X)$ ought to be shunned when convexity in $(\underline{\mathcal{P}})$ is to be promoted, what might be the alternative? This is a serious issue because risk constraints involving this choice are very common, especially in reliability engineering,[74] because

$$q_{\alpha_i}(\underline{f}_i(x)) \le 0 \iff \mathrm{prob}\{\underline{f}_i(x) \le 0\} \ge \alpha_i. \tag{5.3}$$

A strong argument can be made for passing from quantiles/VaR to superquantiles/CVaR by instead taking $\mathcal{R}_i(X) = \bar{q}_{\alpha_i}(X) = \mathrm{CVaR}_{\alpha_i}(X)$. This has the effect of replacing "probability of failure" by an alternative called "buffered probability of failure", which is safer and easier to work with computationally; see [15].

The claim that problem-solving may be easier with CVaR than with VaR could seem surprising from the angle that $\mathrm{CVaR}_\alpha(X)$ is defined as a conditional expectation in a "tail" which is dependent on $\mathrm{VaR}_\alpha(X)$, yet it rests on the characterization in (3.21). But we have explained in [6][75] how, in the case of $(\underline{\mathcal{P}})$ with $\mathcal{R}_i = \mathrm{CVaR}_{\alpha_i}$ for each i, one can expand $\mathrm{CVaR}_{\alpha_i}(\underline{f}_i(x))$ through (3.21) into an expression involving a auxiliary parameter C_i and go on to minimize not only with respect to x but also simultaneously with respect to the C_i's. This has the benefit not only of simplifying the overall minimization but also providing, along with the optimal solution $\bar{x}$ to $(\underline{\mathcal{P}})$, corresponding $\mathrm{VaR}_{\alpha_i}(\underline{f}_i(x))$ values as the optimal $\bar{C}_i$'s.

Now we are in position to point out, on the basis of the risk quadrangle, that this technique has a new and far-reaching extension.

Regret Theorem. *Consider a stochastic optimization problem $(\underline{\mathcal{P}})$ in which each $\mathcal{R}_i$ is a regular measure of risk coming from a regular measure of regret $\mathcal{V}_i$ with associated statistic $\mathcal{S}_i$ by the quadrangle formulas*

$$\mathcal{R}_i(X) = \min_C\{C + \mathcal{V}_i(X - C)\},$$
$$\mathcal{S}_i(X) = \operatorname*{argmin}_C\{C + \mathcal{V}_i(X - C)\}. \tag{5.4}$$

Solving $(\underline{\mathcal{P}})$ can be cast then as solving the expanded problem

$$\text{choose } x = (x_1, \ldots, x_n) \quad \text{and} \quad C_0, C_1, \ldots, C_m \text{ to}$$
$$\text{minimize } C_0 + \mathcal{V}_0(\underline{f}_0(x) - C_0) \quad \text{over } x \in S,\ C_i \in \mathbb{R}, \tag{$\underline{\mathcal{P}}'$}$$
$$\text{subject to } C_i + \mathcal{V}_i(\underline{f}_i(x) - C_i) \le 0 \quad \text{for } i = 1, \ldots, m.$$

An optimal solution $(\bar{x}, \bar{C}_0, \bar{C}_1, \ldots, \bar{C}_m)$ to problem $(\underline{\mathcal{P}}')$ provides as $\bar{x}$ an optimal solution to problem $(\underline{\mathcal{P}})$ and as $\bar{C}_i$ a corresponding value of the statistic $\mathcal{S}_i(\underline{f}_i(\bar{x}))$ for $i = 0, 1, \ldots, m$.

The Mixing Theorem of Section 3 can be combined with Regret Theorem. When $\mathcal{V}_i$ is itself expressed by a minimization formula in extra parameters, these can be brought into $(\underline{\mathcal{P}})$ as well.

The idea behind the Regret Theorem is not restricted to regret measures. It can operate just as well for deviation measures in terms of error measures through the quadrangle principle that

$$\mathcal{D}(X) \le c \iff \mathcal{E}(X - C) \le c \quad \text{for a choice of } C \in \mathbb{R}.$$

Estimation. The topic of generalized regression is next on the agenda. As explained in Section 1, this concerns the approximation of a given random variable Y by a function $f(X_1, \ldots, X_n)$ of other random variables $X_1, \ldots, X_n$. By the regression being "generalized" we mean that the difference $Z_f = Y - f(X_1, \ldots, X_n)$ may be assessed for its nonzeroness by an error measure $\mathcal{E}$ different from the one in "least-squares" as in Example 1, or for that matter even from the kind in quantile regression, as in Example 2. The case of generalized *linear* regression, where the functions f in the approximation are limited to the form

$$f(x_1, \ldots, x_n) = C_0 + C_1x_1 + \cdots + C_nx_n$$
(the linear case), (5.5)

has already been studied in [9], but only for error measures $\mathcal{E}$ that are positively homogeneous. Here we go beyond those limitations and investigate the problem:

$$\text{minimize } \mathcal{E}(Z_f) \text{ over all } f \in \mathcal{C},$$
$$\text{where } Z_f = Y - f(X_1, \ldots, X_n), \tag{5.6}$$

for given random variables $X_1, \ldots, X_n, Y$, and some given class $\mathcal{C}$ of functions f.

Taking $\mathcal{C}$ to be the class in (5.5) with respect to all possible coefficients $C_0, C_1, \ldots, C_n$, would specialize to linear regression, pure and simple. Then $\mathcal{E}(Z_f)$ would be a function of these coefficients and we would be minimizing over $(C_0, C_1, \ldots, C_n) \in \mathbb{R}^{n+1}$. However, even in the linear case there could be further specialization through placing conditions on some of the coefficients, such as perhaps nonnegativity. In fact, a broad example of the kinds of classes of regression functions that can be brought into the picture is the following[76]:

$$\mathcal{C} = \text{all the functions } f(x_1, \ldots, x_n)$$
$$= C_0 + C_1h_1(x_1, \ldots, x_n) + \cdots + C_mh_m(x_1, \ldots, x_n)$$
for given $h_1, \ldots, h_m$ on $\mathbb{R}^n$ and coefficient vectors
$(C_1, \ldots, C_m)$ in a given set $C \subset \mathbb{R}^m$. (5.7)

Motivation for generalized regression comes from applications in which Y has the cost/loss orientation that we have been emphasizing in this paper. Underestimation might then be more dangerous than overestimation, and that could suggest using an asymmetric error measure $\mathcal{E}$, with $\mathcal{E}(Z_f) \ne \mathcal{E}(-Z_f)$.

Further motivation comes from "factor models" and other such regression techniques in finance and engineering, which might have unexpected consequences when utilized in stochastic optimization because of interactions with parameterization by the decision vector x. For instance, if one of the random "costs" $\underline{f}_i(x)$ in problem $(\underline{\mathcal{P}})$ is estimated by such a technique as $\underline{g}_i(x)$, it may be hard to determine the effects this could have on the optimal decision. We have argued in [9], and demonstrated with specific results, that it might be wise to "tune" the regression to the risk measure $\mathcal{R}_i$ applied to $\underline{f}_i(x)$ in $(\underline{\mathcal{P}})$. This would mean passing around the fundamental quadrangle from $\mathcal{R}_i$ to an error measure $\mathcal{E}_i$ in the same quartet.

Regression Theorem. *Consider problem (5.6) for random variables $X_1, \ldots, X_n$ and Y in the case of $\mathcal{E}$ being a regular measure of error and $\mathcal{C}$ being a class of functions $f : \mathbb{R}^n \to \mathbb{R}$ such that*

$$f \in \mathcal{C} \Longrightarrow f + C \in \mathcal{C} \quad \text{for all } C \in \mathbb{R}. \tag{5.8}$$

Let $\mathcal{D}$ and $\mathcal{S}$ correspond to $\mathcal{E}$ as in the Quadrangle Theorem. Problem (5.6) is equivalent then to:

$$\text{minimize } \mathcal{D}(Z_f) \text{ over all } f \in \mathcal{C} \quad \text{such that } 0 \in \mathcal{S}(Z_f), \tag{5.9}$$

74 The article [39] furnishes illuminating background.

75 See also the tutorial paper [30].

76 It should also be kept in mind that a possibly nonlinear change of scale in the variables, such as passing to logarithms, could be executed prior to this depiction.

which in the case of a class $\mathcal{C}$ as in (5.7) and $H_i = h_i(X_1, \ldots, X_n)$ comes down to

$$\begin{aligned} &\text{minimize } \mathcal{D}(Y - [C_1H_1 + \cdots + C_mH_m]), \\ &\quad \text{then take } C_0 \in \mathcal{S}(Y - [C_1H_1 + \cdots + C_mH_m]). \end{aligned} \tag{5.10}$$

Moreover if $\mathcal{E}$ is of expectation type and $\mathcal{C}$ includes a function f satisfying

$$\begin{aligned} &f(x_1, \ldots, x_n) \in \mathcal{S}(Y(x_1, \ldots, x_n)) \\ &\quad \text{almost surely for } (x_1, \ldots, x_n) \in D, \\ &\text{where } Y(x_1, \ldots, x_n) = Y_{X_1 = x_1, \ldots, X_n = x_n} \\ &\quad \text{(conditional distribution)}, \end{aligned} \tag{5.11}$$

with D being the support of the distribution in $\mathbb{R}^n$ induced by $X_1, \ldots, X_n$,[77] *then that f solves the regression problem and tracks this conditional statistic*[78] *in the sense that*

$$f(X_1, \ldots, X_n) \in \mathcal{S}(Y(X_1, \ldots, X_n)) \text{ almost surely}. \tag{5.12}$$

The first part of this result generalizes [9, Theorem 3.2] on linear regression through elementary extension of the same proof. The specialization in (5.10) relies on $\mathcal{D}(Z - C_0) = \mathcal{D}(Z)$ and $\mathcal{S}(Z - C_0) = \mathcal{S}(Z) - C_0$. The second part is new. It comes from the observation that, in the expectation case, if f satisfies (5.11), then for any other $g \in \mathcal{C}$ one has

$$\begin{aligned} &\mathcal{E}(Y(x_1, \ldots, x_n) - f(x_1, \ldots, x_n)) \\ &\quad \le \mathcal{E}(Y(x_1, \ldots, x_n) - g(x_1, \ldots, x_n)) \\ &\text{almost surely for } (x_1, \ldots, x_n) \in D. \end{aligned}$$

When $\mathcal{E}$ is of expectation type, this inequality can be "integrated" over the distribution of $(X_1, \ldots, X_n)$ to obtain $\mathcal{E}(Y(X_1, \ldots, X_n) - f(X_1, \ldots, X_n)) \le \mathcal{E}(Y(X_1, \ldots, X_n) - g(X_1, \ldots, X_n))$.

Apart from that special circumstance, the question of the existence of an optimal regression function $f \in \mathcal{C}$ has not been addressed in the theorem, because we are reluctant in the present context to delve deeply into the possible structure of the class $\mathcal{C}$. But existence in the case of linear regression has been covered in [9, Theorem 3.1], and similar considerations would apply to the broader class in (5.7), with the coefficient set C taken to be closed.[79]

There could be many applications of these ideas, and much remains to be explored and developed. Some related research in special cases, largely concerned with quantile regression, can be seen in [40–42]; see also [39] for further motivation, and moreover [43].

The measure of error in quantile regression is indeed of expectation type, so that the second part of our Regression Theorem can be applied if the class $\mathcal{C}$ of functions f is rich enough. The class of linear functions of $X_1, \ldots, X_n$ would very likely not meet that standard, but the class in (5.7) may offer hope through judicious choice of $h_1, \ldots, h_m$.

[77] Almost surely, in (5.11), refers to this distribution.

[78] It is assumed, for this part, that the distribution of $Y(x_1, \ldots, x_n)$ for $(x_1, \ldots, x_n) \in D$ belongs to $\mathcal{L}^2(\Omega)$, and the same then for the random variable $Y(X_1, \ldots, X_n)$ obtained from it.

[79] Work with the class in (5.7), which does of course satisfy (5.9), can actually be reduced to the linear case, so that generalized linear theory can be applied. To do this we can introduce new random variables $W_i = h_i(X_1, \ldots, X_n)$ with distributions inherited from the X_j's and carry out linear regression of Y with respect to $W_1, \ldots, W_m$.

6. Probability modeling and the dualization of risk

More explanation about the view of uncertainty that we take here may be helpful, especially for the sake of those who would like to make use of the ideas without having to go too far into the technical mathematics of probability theory. In modeling uncertain quantities as random variables, we tacitly regard them as having probability distributions, but this does not mean we assume those distributions are directly known. Sampling, for instance, might be required to learn more, and even then, only approximations might be available.

The characteristics of a random variable X, by itself, are embodied in its cumulative distribution function F_X, with $F_X(x) = \text{prob}\{X \le x\}$. This induces a probability measure on the real numbers $\mathbb{R}$ which may or may not be expressible by a density function f with respect to ordinary integration, i.e., as $dF_X(x) = f(x)dx$. The lack of a density function is paramount when X is a discrete random variable with only finitely many possible outcomes. Then F_X is a step function.

Sometimes the underlying uncertainty being addressed revolves around observations of several random variables $V_1, \ldots, V_m$, and their *joint* distribution. The corresponding probability measure on $\mathbb{R}^m$ is induced then by the multivariate distribution function

$$\begin{aligned} &F_{V_1, \ldots, V_m}(v_1, \ldots, v_m) \\ &\quad = \text{prob}\{(V_1, \ldots, V_m) \le (v_1, \ldots, v_m)\}. \end{aligned} \tag{6.1}$$

Functions $x = g(v_1, \ldots, v_m)$ give rise to random variables $X = g(V_1, \ldots, V_m)$ having $F_X(x) = \text{prob}\{g(V_1, \ldots, V_m) \le x\}$. Again, the distribution of $(V_1, \ldots, V_m)$ need not be describable by a density function $f(v_1, \ldots, v_m)$. We might be dealing with a discrete distribution of $(V_1, \ldots, V_m)$ corresponding to an m-dimensional "scatter plot".

The standard framework of a *probability space* serves for handling all these aspects of randomness easily and systematically. It consists of a set Ω supplied with a probability measure P_0 and a field $\mathcal{A}$ of its subsets.[80] We think of the elements $\omega \in \Omega$ as "future states" (of information), or "scenarios". Having a subset A of Ω belong to $\mathcal{A}$ means that the probability of ω being in A is regarded as known in the present: $\text{prob}\{A\} = P_0(A)$. In that way, the field $\mathcal{A}$ is a model for present information about the future. There could be multistage approaches to such information, in which $\mathcal{A}$ is just the first in a chain of ever-larger collections of subsets of Ω, but we are not looking at that. A scenario ω could, in our setting, nonetheless involve multiple time periods, but we are not going to consider, here, how additional observations, as the scenario unfolds, might be put to use in optimization.

Random variables in this framework are functions $X : \Omega \to \mathbb{R}$, with future outcomes $X(\omega)$, such that, for every $x \in \mathbb{R}$, the set $A = \{\omega \mid X(\omega) \le x\}$ belongs to $\mathcal{A}$.[81] The expected value of a random variable X is the integral $EX = \int_\Omega X(\omega) dP_0(\omega)$. As a special case, the probability space $(\Omega, \mathcal{A}, P_0)$ could be generated by future observations of some variables $V_1, \ldots, V_m$, as above, in which case Ω would be a subset of $\mathbb{R}^m$ with elements $\omega = (v_1, \ldots, v_m)$ and P_0 would be the probability measure induced by the joint distribution function $F_{V_1, \ldots, V_m}$. If P_0 has a density function $f(v_1, \ldots, v_m)$ with respect to ordinary integration, then for $X = g(V_1, \ldots, V_m)$ one has $EX = \int g(v_1, \ldots, v_m) f(v_1, \ldots, v_m) dv_1 \cdots dv_m$, but without such a density, it is not possible to rely this way on $dv_1 \cdots dv_m$. That is why, in achieving adequate generality, it is crucial to refer

[80] We write P_0 for this underlying probability measure in order to reserve P for general purposes below.

[81] The sets $A \in \mathcal{A}$ are called the "measurable" sets and the functions X in question the "measurable" functions.

to a background probability measure P_0 as the source of all the distributions that come up.

Despite that focus, a means is provided for considering alternatives P to P_0, and indeed this will be very important in subsequent discussions of risk and its dualization. Other probability measures P can enter the picture as long as the expected value $E_P(X) = \int X(\omega)dP(\omega)$ can be expressed by $E[XQ] = \int X(\omega)Q(\omega)dP_0(\omega)$ for some random variable Q, which is then called *the density of P with respect to P_0* with notation $Q = dP/dP_0$.[82] For instance, in the case where Ω has finitely many elements ω_k, $k = 1, \ldots, N$, if P_0 gives them equal weight $1/N$ but P assigns probability p_k to ω_k, then $Q(\omega_k) = p_k N$.

Another point needing emphasis is that little is really lost in supposing the existence of an underlying probability measure P_0, even if prospects of knowing much about it are low. Convenience in theory can be served nonetheless. In "robust optimization", for example, direct probability is in principle avoided, and yet a so-called *uncertainty set* has to be constructed. That set, often identified through rough considerations of probability anyway, can be identified here with the space Ω. The worst-case risk measure $\mathcal{R}(X) = \sup X$, which is the prime focus of "robust optimization", is captured anyway as generated by considering all P alternative to P_0 in the above sense, as will be explained below. Similarly, the "distributed worst-case" risk measure of Example 6 is covered without having to know very much about P_0.

The need to deal securely with expectations of random variables and certain products of random variables forces some restrictions. For any random variable X, the expressions $\|X\|_p$ introduced earlier are well defined but could be ∞. It is common practice to work with the spaces[83]

$$\mathcal{L}^p(\Omega) = \mathcal{L}^p(\Omega, \mathcal{A}, P_0) = \left\{X \mid \|X\|_p < \infty\right\},$$
$$\text{where } \mathcal{L}^1(\Omega) \supset \cdots \supset \mathcal{L}^p(\Omega) \supset \cdots \supset \mathcal{L}^\infty(\Omega). \tag{6.2}$$

For any X in these spaces, EX is well defined and finite, but the situation for products of random variables, like XQ above, is more delicate. While there are options with X in one space and Q in another, no choice is perfect.

For our purposes here, $\mathcal{L}^2(\Omega)$ has been taken as the platform. That has the simplifying advantage that $E|XQ| < \infty$ for any $X \in \mathcal{L}^2(\Omega)$ and $Q \in \mathcal{L}^2(\Omega)$. However, it does mean that, in considering alternative probability measures P with densities $Q = dP/dP_0$ the restriction must be made to the cases where $\int_\Omega (dP/dP_0)^2(\omega)dP_0 < \infty$. Actually, though, this restriction makes little difference in the end, because other probability measures can adequately be mimicked by these (and for finite Ω is no restriction at all).

Dualization concerns the development of "dual representations" of various functionals, also called "envelope representations", which can yield major insights and provide tools for characterizing optimality. The functionals $\mathcal{F}$ may in general take on ∞ as a value (although usually $-\infty$ is excluded), and some notation for handling that is needed. The *effective domain* of $\mathcal{F}$ is the set

$$\operatorname{dom}\mathcal{F} = \{X \in \mathcal{L}^2(\Omega) \mid \mathcal{F}(X) < \infty\}. \tag{6.3}$$

When $\mathcal{F}$ is convex, this set is convex, but $\mathcal{F}$ closed convex does not necessarily entail $\operatorname{dom}\mathcal{F}$ also being closed. The platform for dualization is a correspondence for closed convex functionals $\mathcal{F}$:[84,85]

$$\begin{aligned}&\mathcal{F} : \mathcal{L}^2 \to (-\infty, \infty] \text{ closed convex,}\\ &\mathcal{F} \neq \infty \Longleftrightarrow \exists \mathcal{G} : \mathcal{L}^2 \to (-\infty, \infty], \quad \mathcal{G} \not\equiv \infty, \quad \text{with}\\ &\mathcal{F}(X) = \sup_{Q \in \mathcal{L}^2(\Omega)} \{E[XQ] - \mathcal{G}(Q)\} \quad \text{for all } X.\\ &\text{Moreover the lowest such } \mathcal{G} \text{ is } \mathcal{G} = \mathcal{F}^*,\\ &\quad \text{where } \mathcal{F}^* \text{is closed convex and given by}\\ &\mathcal{F}^*(Q) = \sup_{X \in \mathcal{L}^2(\Omega)} \{E[XQ] - \mathcal{F}(X)\} \quad \text{for all } Q.\end{aligned} \tag{6.4}$$

The functional $\mathcal{F}^*$ is said to be conjugate to $\mathcal{F}$, which in turn is conjugate to $\mathcal{F}^*$ through the first formula in (6.4) in the case of $\mathcal{G} = \mathcal{F}^*$, namely

$$\mathcal{F}(X) = \sup_{Q \in \mathcal{L}^2(\Omega)} \{E[XQ] - \mathcal{F}^*(Q)\} \quad \text{for all } X. \tag{6.5}$$

The nonempty convex set $\operatorname{dom}\mathcal{F}^* = \{Q \mid \mathcal{F}^*(Q) < \infty\}$ can replace $\mathcal{L}^2(\Omega)$ in this formula, and similarly $\operatorname{dom}\mathcal{F}$ can replace $\mathcal{L}^2(\Omega)$ in the first formula of (6, 4). Here are some cases that will be especially important to us[86]:

$$\text{for } \mathcal{F} \text{ closed convex } \not\equiv \infty :$$
$$\begin{cases} \mathcal{F}(0) = 0 \Longleftrightarrow \inf \mathcal{F}^* = 0,\\ \mathcal{F}(X) \geq EX \Longleftrightarrow \mathcal{F}^*(1) \leq 0,\\ \mathcal{F} \text{ is monotonic} \Longleftrightarrow Q \geq 0 \\ \quad \text{when } Q \in \operatorname{dom}\mathcal{F}^*,\\ \mathcal{F} \text{ is pos. homog.} \Longleftrightarrow \mathcal{F}^*(Q) = 0 \\ \quad \text{when } Q \in \operatorname{dom}\mathcal{F}^*, \end{cases} \tag{6.6}$$

(where the "1" in the second line refers to the constant r.v. with value 1). The final case, with positive homogeneity, says that

there is a one-to-one correspondence between nonempty,
closed, convex sets $\mathcal{Q} \subset \mathcal{L}^2(\Omega)$
and closed convex pos. homogeneous functionals

$$\begin{aligned}&\mathcal{F} : \mathcal{L}^2 \to (-\infty, \infty], \text{ given by}\\ &\mathcal{F}(X) = \sup_{Q \in \mathcal{Q}} E[XQ] \quad \text{for all } X, \text{ where}\\ &\mathcal{Q} = \{Q \mid E[XQ] \leq \mathcal{F}(X) \text{ for all } X\}.\end{aligned} \tag{6.7}$$

The second formula in (6.7) identifies $\mathcal{Q}$ with $\operatorname{dom}\mathcal{F}^*$. Any $\mathcal{Q}$ for which the first formula holds must moreover have $\operatorname{dom}\mathcal{F}^*$ as its closed, convex hull.

Envelope Theorem.[87] *The functionals $\mathcal{J}$ that are the conjugates $\mathcal{R}^*$ of the regular measures of risk $\mathcal{R}$ on $\mathcal{L}^2(\Omega)$ are the closed convex functionals $\mathcal{J}$ with effective domains $\mathcal{Q} = \operatorname{dom}\mathcal{J}$ such that*

(a) $EQ = 1$ *for all* $Q \in \mathcal{Q}$,
(b) $0 = \mathcal{J}(1) \leq \mathcal{J}(Q)$ *for all* $Q \in \mathcal{Q}$,

[82] Such measures P are said to be "absolutely continuous" with respect to P_0.

[83] When Ω is a discrete set of N elements, these spaces coincide and can be identified with $\mathbb{R}^N$.

[84] See Theorem 5 of [27]; this is the case of $\mathcal{L}^2(\Omega)$ paired with itself through $\langle X, Q\rangle = E[XQ]$. The operation $\mathcal{F} \to \mathcal{F}^*$ is called the Legendre–Fenchel transform.

[85] Saying $\mathcal{F}^*$ is "lowest" means here that every $\mathcal{G}$ with the indicated property satisfies $\mathcal{G}(Q) \geq \mathcal{F}^*(Q)$ for all $Q \in \mathcal{L}^2$.

[86] The first is immediate from (6.5) with $X = 1$, while the second follows from (6.4) with $Q = 1$. In the third, the sufficiency comes from (6.5), and the necessity as well, because monotonicity of $\mathcal{F}$ precludes the existence of a nonmonotonic affine functional $\mathcal{L}$ with $\mathcal{L}(X) \leq \mathcal{F}(X)$ for all X. The necessity in the fourth is clear from (6.4) (because positive homogeneity allows only 0 or ∞ as the supremum); the sufficiency is obvious from (6.5).

[87] Most of the facts in this compilation, which follow from the general properties of conjugacy as above, are already well understood and have been covered, for instance, in [10]. The new aspects are the dualization of aversity in condition (c) and the final assertion, connecting with the dualization of regret.

(c) *for each nonconstant* $X \in \mathcal{L}^2(\Omega)$ *there exists* $Q \in \mathcal{Q}$ *such that* $E[XQ] - EX > \mathcal{J}(Q)$.

The dual representation of $\mathcal{R}$ *corresponding to* $\mathcal{J} = \mathcal{R}^*$ *is*

$$\mathcal{R}(X) = \sup_{Q\in\mathcal{Q}} \{E[XQ] - \mathcal{J}(Q)\}. \tag{6.8}$$

Here $\mathcal{R}$ *is positively homogeneous if and only if* $\mathcal{J}(Q) = 0$ *for all* $Q \in \mathcal{Q}$, *whereas* $\mathcal{R}$ *is monotonic if and only if* $Q \geq 0$ *for all* $Q \in \mathcal{Q}$.

If $\mathcal{V}$ *is a regular measure of regret that projects to* $\mathcal{R}$, *then* $\mathcal{Q} = \{Q \in \operatorname{dom}\mathcal{V}^* \mid EQ = 1\}$ *and the conjugate* $\mathcal{J} = \mathcal{R}^*$ *has* $\mathcal{J}(Q) = \mathcal{V}^*(Q)$ *for* $Q \in \mathcal{Q}$, *but* $\mathcal{J}(Q) = \infty$ *for* $Q \notin \mathcal{Q}$.

The error measure $\mathcal{E}$ *paired with the regret measure* $\mathcal{V}$ *has* $\mathcal{E}^*(X) = \mathcal{V}^*(X+1)$. *Likewise, the deviation measure* $\mathcal{D}$ *paired with the risk measure* $\mathcal{R}$ *has* $\mathcal{D}^*(X) = \mathcal{R}^*(X+1)$.

Risk envelopes and identifiers. The convex set $\mathcal{Q}$ in this theorem is called the *risk envelope* associated with $\mathcal{R}$, and a Q furnishing the maximum in (6.8) is a *risk identifier* for X.

The monotonic case in the theorem combines $EQ = 1$ with $Q \geq 0$ and thereby allows us to interpret each $Q \in \mathcal{Q}$ as a probability density dP/dP_0 describing an alternative probability measure P on Ω. For positively homogeneous $\mathcal{R}$, the $\mathcal{J}(Q)$ term drops out of the representation in (6.8) (by being 0). The formula then characterizes $\mathcal{R}(X)$ as giving the worst "cost" that might result from considering the expected values $E[XQ] = E_P[X]$ over all those alternative probability measures P having densities Q in the risk envelope $\mathcal{Q}$.

The nonhomogeneous case has a similar interpretation, but distinguishes within $\mathcal{Q}$ a subset $\mathcal{Q}_0$ consisting of the densities Q for which $\mathcal{J}(Q) = 0$, which always includes $Q \equiv 1$ (the density of P_0 with respect to itself). Densities Q that belong to $\mathcal{Q}$ but not $\mathcal{Q}_0$ have $\mathcal{J}(Q) \in (0, \infty)$. In (6.8) that term then drags the expectation down. In a sense, $\mathcal{J}(Q)$ downgrades the importance of such densities.

The conjugates $\mathcal{V}^*$ of regular measures of regret $\mathcal{V}$ have virtually the same characterization as the conjugates $\mathcal{R}^*$ in the theorem. Property (a) is omitted, but on the other hand there is a provision to enforce the property in (3.15) (in the cases when it is not guaranteed to hold automatically). This provision is that $\mathcal{V}^*(C) < \infty$ for C near enough to 1.

Some examples of risk envelopes in the *positively homogeneous* case, where (6.8) holds with $J(Q)$ omitted, are the following[88]:

$$\begin{aligned}
&\mathcal{R}(X) = EX + \lambda\sigma(X) \longleftrightarrow \\
&\qquad \mathcal{Q} = \{Q = 1 + \lambda Y \mid \|Y\|_2 \leq 1, EY = 0\} \\
&\mathcal{R}(X) = \mathrm{CVaR}_\alpha(X) \longleftrightarrow \\
&\qquad \mathcal{Q} = \left\{Q \mid 0 \leq Q \leq \frac{1}{1-\alpha}, EQ = 1\right\} \\
&\mathcal{R}(X) = \sup X \longleftrightarrow \mathcal{Q} = \{Q \mid Q \geq 0, EQ = 1\} \\
&\mathcal{R}(X) = \sum_{k=1}^{r} \lambda_k \mathcal{R}_k(X) \longleftrightarrow \\
&\qquad \mathcal{Q} = \left\{Q = \sum_{k=1}^{r} \lambda_k Q_k \mid Q_k \in \mathcal{Q}_k\right\}, \\
&\quad \text{where } \mathcal{R}_k \longleftrightarrow \mathcal{Q}_k.
\end{aligned} \tag{6.9}$$

Another illustration comes out of Example 6, which can now be formalized via (2.5) in terms of a partition of Ω into disjoint subsets Ω_k of probability $p_k > 0$ with $\sup_k X$ being the essential supremum of X on Ω_k and $E_k X$ being the conditional expectation $E[X|\Omega_k]$:

$$\begin{aligned}
&\mathcal{R}(X) = \sum_{k=1}^{r} p_k \sup_k X \longleftrightarrow \\
&\qquad \mathcal{Q} = \left\{Q \geq 0 \mid p_k = E[Q|\Omega_k] = \int_{\Omega_k} Q(\omega) dP_0(\omega)\right\}.
\end{aligned} \tag{6.10}$$

The risk envelope $\mathcal{Q}$ for the p-order superquantile risk measure of Example 12 has not specifically been worked out, but strong clues have been furnished by Dentcheva et al. [25]. The dual expression derived there indicates that the risk envelope in this case is a union of risk envelopes for mixed quantile risk measures like (2.7) (which are covered by the second and fourth cases of (6.9), except that finite sums need to be replaced by general "continuous" sums as in (2.8)).

Examples beyond positive homogeneity, where nonzero values of $\mathcal{J}$ may enter, are simple to work out in the expectation case:

For quadrangles in the Expectation Theorem, with regret $\mathcal{V}(X) = E[v(X)]$, the conjugate $\mathcal{J} = \mathcal{R}^*$ of the risk measure $\mathcal{R}$ projected from $\mathcal{V}$ is given by

$$\mathcal{J}(Q) = \begin{cases} E[v^*(Q)] & \text{if } EQ = 1 \\ \infty & \text{if } EQ \neq 1 \end{cases} \text{ for the function } v^* \text{ conjugate to } v, \tag{6.11}$$

given by $v^*(q) = \sup_x \{xq - v(x)\}$.

The properties of v^* corresponding to those of v in (4.3) are that v^* is closed convex with $v^*(1) = 0$, $v^{*\prime}(1) = 0$.

This holds from the description in Envelope Theorem of the $\mathcal{J}$ in projection from $\mathcal{V}$ because the functional conjugate to $\mathcal{V}(X) = E[v(X)]$ is $\mathcal{V}^*(Q) = E[v^*(Q)]$.[89] The dualization of the properties of v to those of v^* comes from one-dimensional convex analysis; see [26].

An especially interesting illustration is furnished by Example 8, where one has

$$v(x) = \exp x - 1, \qquad v^*(q) = \begin{cases} q\log q - q & \text{if } q \geq 0, \\ \infty & \text{if } q < 0, \end{cases} \tag{6.12}$$

with $0 \log 0 = 0$, the usual convention. Through (6.11) this yields

$$\begin{aligned}
&\mathcal{R}(X) = \log E[\exp X] \longleftrightarrow \\
&\qquad \mathcal{J}(Q) = \begin{cases} E[Q\log Q] & \text{if } Q \geq 0, EQ = 1, \\ \infty & \text{otherwise.} \end{cases}
\end{aligned} \tag{6.13}$$

Here $-\mathcal{J}(Q)$ is a well known expression for the *relative entropy* with respect to the probability measure P_0 of the probability measure P having $Q = dP/dP_0$.[90]

Results in [27] can exploit the general dualization of $\mathcal{R}$ to $\mathcal{J}$ through Lagrangian formats for optimization involving $\mathcal{R}$ which generate *dual problems*. Even more powerful developments of optimization duality, tailored to the fine points of financial mathematics, have recently been contributed by Pennanen [44]. For more insights on entropic modeling versus risk, see [4], which emphasizes the role of deviation measures $\mathcal{D}$ in place of risk measures $\mathcal{R}$.

Also coming out of the Envelope Theorem is further insight into the degree of nonuniqueness of the error measures $\mathcal{E}$ that project to a specified deviation measure $\mathcal{D}$, or the regret measures $\mathcal{V}$ that project to a specified risk measure $\mathcal{R}$. In the positively homogeneous case of $\mathcal{V}$, for instance, the conjugate $\mathcal{V}^*$ has by (6.6) and (6.7), the simple form that it is 0 on a certain closed, convex set $\mathcal{K}$ but ∞ outside of $\mathcal{K}$; then $\operatorname{dom}\mathcal{V}^* = \mathcal{K}$. The theorem says the risk envelope $\mathcal{Q}$ determining the risk measure $\mathcal{R}$ projected from $\mathcal{V}$ has $\mathcal{Q}$ equal to the intersection of $\mathcal{K}$ with the hyperplane $\{Q \mid EQ = 1\}$. That intersection only uses one "slice" of $\mathcal{K}$. Different $\mathcal{K}$'s that agree for this "slice" will give different $\mathcal{V}$'s yielding the same $\mathcal{R}$. Discovering a "natural" antecedent $\mathcal{V}$ for $\mathcal{R}$ therefore

[88] These envelopes were worked out in [22]; see also [21].

[89] This follows a general rule of convex analysis in [27, Theorem 21]. The "inner product" in the function space $\mathcal{L}^2(\Omega)$ is $\langle X, Q\rangle = E[XQ]$.

[90] See [7] for more background. Another name for this is Kullback–Leibler distance.

amounts geometrically to discovering a "natural" extension $\mathcal{K}$ of $\mathcal{Q}$ beyond the hyperplane $\{Q \mid EQ = 1\}$.

The Envelope Theorem, as presented here, is based on duality theory in convex analysis, but the idea of expressing preferences through functionals defined by a max or min over a set of probability measures, as a representation of distrust or ambiguity, is far from new. In finance, the concept is often attributed to Artzner et al. [1], but in statistics it can be traced to Huber [45] and his *sublinear expectation* functionals. There is a strong echo also in the theory of preferences in economics, where a minimum of expected utility over a set of probability measures has been explored from various angles. For that literature, see [46,47], and their references.

References

[1] P. Artzner, F. Delbaen, J.-M. Eber, D. Heath, Coherent measures of risk, Mathematical Finance 9 (1999) 203–227.
[2] R.T. Rockafellar, S. Uryasev, M. Zabarankin, Master funds in portfolio analysis with general deviation measures, Journal of Banking and Finance 30 (2) (2006) 743–778.
[3] R.T. Rockafellar, S. Uryasev, M. Zabarankin, Optimality conditions in portfolio analysis with general deviation measures, Mathematical Programming, Series B 108 (2006) 515–540.
[4] B. Grechuk, A. Molyboha, M. Zabarankin, Maximum entropy principle with general deviation measures, Mathematics of Operations Research 34 (2009) (2008) 445–467.
[5] R.T. Rockafellar, S. Uryasev, Optimization of conditional value-at-risk, Journal of Risk 2 (2000) 21–42.
[6] R.T. Rockafellar, S. Uryasev, Conditional value-at-risk for general loss distributions, Journal of Banking and Finance 26 (2002) 1443–1471.
[7] A. Ben Tal, M. Teboulle, An old–new concept of convex risk measures: the optimal certainty equivalent, Mathematical Finance 17 (2007) 449–476.
[8] P.A. Krokhmal, Higher moment coherent risk measures, Quantitative Finance 7 (2007) 373–387.
[9] R.T. Rockafellar, S. Uryasev, M. Zabarankin, Risk tuning with generalized linear regression, Mathematics of Operations Research 33 (3) (2008) 712–729.
[10] H. Föllmer, A. Schied, Stochastic Finance, second ed., De Gruyter, New York, 2004.
[11] T. Gneiting, Making and evaluating point forecasts, Journal of the American Statistical Association 106 (2011) 746–762.
[12] R. Koenker, G. Bassett, Regression quantiles, Econometrica 46 (1978) 33–50.
[13] C. Acerbi, Spectral measures of risk: a coherent representation of subjective risk aversion, Journal of Banking and Finance 26 (2002) 1505–1518.
[14] R.S. Dembo, A.J. King, Tracking models and the optimal regret distribution in asset allocation, Applied Stochastic Models 8 (1992) 151–157.
[15] R.T. Rockafellar, J.O. Royset, On buffered failure probability in design and optimization of structures, Journal of Reliability Engineering and System Safety 99 (2010) 499–510.
[16] D.G. Luenberger, Investment Science, Oxford University Press, 1998.
[17] M.E. Yaari, The dual theory of choice under risk, Econometrica 55 (1987) 95–115.
[18] A. Roell, Risk aversion in Quiggin and Yaari's rank-order model of choice under uncertainty, The Economic Journal, Issue Supplement: Conference papers 97 (1987) 143–159.
[19] D. Dentcheva, A. Ruszczynski, Common mathematical foundations of expected utility and dual utility theories, SIAM Journal on Optimization 24 (2013).
[20] G. Pflug, On distortion functionals, Stastistics and Decisions 27 (2009) 201–209.
[21] R.T. Rockafellar, S. Uryasev, M. Zabarankin, Generalized deviations in risk analysis, Finance and Stochastics 10 (2006) 51–74.
[22] R.T. Rockafellar, S. Uryasev, M. Zabarankin, Deviation measures in risk analysis and optimization, Technical Report 2002-7, Department of Industrial and Systems Engineering, University of Florida, 2002.
[23] R.T. Rockafellar, J.O. Royset, Random variables, monotone relations, and convex analysis, Mathematical Programming B (2013) (forthcoming).
[24] Optimization and Risk Management Case Studies with Portfolio SafeGuard (PSG), in: AORDA—American Optimal Decisions, 2010, See Mortgage Pipeline Hedging.
[25] D. Dentcheva, S. Penev, A. Ruszczynski, Kusuoka representation of higher-order dual risk measures, Annals of Operations Research (2013).
[26] R.T. Rockafellar, Convex Analysis, Princeton University Press, Princeton, NJ, 1970.
[27] R.T. Rockafellar, Conjugate Duality and Optimization, in: The Conference Board of Math. Sciences Series, No. 16, SIAM, Philadelphia, 1974.
[28] A. Ruszczyński, A. Shapiro, Optimization of convex risk functions, Mathematics of Operations Research 31 (2006) 433–452.
[29] F. Delbaen, Coherent risk measures on general probability spaces, in: Advances in Finance and Stochastics, Essagys in Honour of Dieter Sondermann, Springer-Verlag, Berlin, 2002, pp. 1–37.
[30] R.T. Rockafellar, Coherent approaches to risk in optimization under uncertainty, in: Tutorials in Operations Research INFORMS 2007, 2007, pp. 38–61.
[31] G. Pflug, Some remarks on the value-at-risk and the conditional value-at-risk, in: S. Uryasev (Ed.), Probabilistic Constrained Optimization: Methodology and Applications, Kluwer Academic Publishers, Norwell, MA, 2000.
[32] C. Acerbi, D. Tasche, On the coherence of expected shortfall, Journal of Banking and Finance 26 (2002) 1487–1503.
[33] R. Koenker, Quantile Regression, in: Econometric Society Monograph Series, Cambridge University Press, 2005.
[34] A. Ben Tal, M. Teboulle, Portfolio theory for the recourse certainty equivalent maximizing investor, Annals of Operations Research 31 (1991) 479–499.
[35] A. Ben Tal, A. Ben-Israel, A recourse certainty equivalent for decisions under uncertainty, Annals of Operations Research 30 (1991) 3–44.
[36] A. Ben Tal, A. Ben-Israel, Duality and equilibrium prices in economics of uncertainty, Mathematical Methods of Operations Research 46 (1997) 51–85.
[37] P. Barrieu, N. El Karoui, Inf-convolution of risk measures and optimal risk transfer, Finance and Stochastics 9 (2005) 269–298.
[38] D. Kahneman, A. Tversky, Prospect theory: an analysis of decision under risk, Econometrica 57 (1979) 263–291.
[39] S. Samson, S. Thoomu, G. Fadel, J. Reneke, Reliable design optimization under aleatory and epistemic uncertainties, in: Proceedings of IDETC/DAC 2009, ASME 2009 International Design Engineering Technical Conferences & 36th Design Automation Conference, San Diego, California, August 30–September 2, 2009.
[40] A. Trindade, S. Uryasev, Improved tolerance limits by combining analytical and experimental data: an information integration methodology, The Journal of Data Science 4 (2006) (2005) 371–386.
[41] A. Trindade, S. Uryasev, Optimal determination of percentiles and allowables: CVaR regression approach, in: A.J. Kurdila, et al. (Eds.), Robust Optimization-Directed Design, Springer Publishers, 2006, pp. 179–247.
[42] A. Golodnikov, Y. Macheret, A. Trindate, S. Uryasev, G. Zrazhevski, Statistical modelling of composition and processing parameters for alloy development: a statistical model-based approach, Journal of Industrial and Management Optimization 3 (2007) (view online).
[43] A. Trindade, S. Uryasev, A. Shapiro, G. Zrazhevsky, Financial prediction with constrained tail risk, Journal of Banking and Finance 31 (2007) 3524–3538.
[44] T. Pennanen, Convex duality in stochastic programming and mathematical finance, Mathematics of Operations Research 36 (2011) 340–362.
[45] P. Huber, Robust Statistics, Wiley, 1981.
[46] P. Maccheroni, M. Marinacci, A. Rustichini, Ambiguity aversion, robustness, and the variational representation of preferences, Econometrica 74 (2006) 1447–1498.
[47] T. Strzalecki, Axiomatic foundations of multiplier preferences, Econometrica 79 (2011) 47–73.