Advice for Mathematically Rigorous Bounds on Optima of Linear Programs

Jared T. Guilbeau
jtg7268@louisiana.edu

Md. I. Hossain:
mhh9786@louisiana.edu

S. D. Karhbet:
sdk6173@louisiana.edu

R. B. Kearfott:
rbk@louisiana.edu

T. S. Sanusi:
tss9743@louisiana.edu

Lihong Zhao:
lxz6134@louisiana.edu

University of Louisiana, P.O. Box 4-1010, Lafayette, Louisiana 70504-1010 USA

October 30, 2016

Abstract

We discuss problems we have encountered when implementing algorithms and formulas for computing mathematically rigorous bounds on the optima of linear programs. The rigorous bounds are based on a 2004 idea of Neumaier and Shcherbina. In a note in the *Journal of Global Optimization*, we pointed out two minor, correctable but consequential and hard to find errors in the original 2004 article. Here, we discuss separate problems implementers may make if they do not take care when interpreting results that are supposed to be mathematically rigorous, or when they reformulate particular problems. Along these lines, we present a catalog of formulas, derived using the Neumaier / Shcherbina idea, from which practitioners may choose when implementing mathematically rigorous bound computations for specific classes of linear programs.

This report is meant to be a companion to [4], where we corrected certain hard-to-find minor but consequential errors in the seminal paper [5] by Neumaier and Shcherbina. We first review the corrected formulas from the Neumaier and Shcherbina. We then point out items of caution for implementers. Finally, we give a catalog of variations of the original Neumaier / Shcherbina formulas.

1 The Neumaier / Shcherbina Formulas

The first formula in [5] is

$$\text{Primal:} \quad \begin{cases} \text{minimize} & c^T x, & c \text{ and } x \in \mathbb{R}^n \\ \text{subject to} & A_{\mathrm{e}} x = b_{\mathrm{e}}, & A_{\mathrm{e}} \in \mathbb{R}^{m_{\mathrm{e}} \times n},\ b_{\mathrm{e}} \in \mathbb{R}^{m_{\mathrm{e}}}, \\ & 0 \le x. & \end{cases}$$

$$\text{Dual:} \quad \begin{cases} \text{maximize} & b_{\mathrm{e}}^T y, \\ \text{subject to} & A_{\mathrm{e}}^T y \le c. \end{cases} \tag{1}$$

The most general formula in [5] is is

$$\text{Primal:} \quad \begin{cases} \text{minimize} & c^T x, & c \text{ and } x \in \mathbb{R}^n \\ \text{subject to} & \underline{b} \le Ax \le \overline{b}, & A \in \mathbb{R}^{m \times n},\ \underline{b}, \overline{b} \in \mathbb{R}^m. \end{cases}$$

$$\text{Dual:} \quad \begin{cases} \text{maximize} & \underline{b}^T y - \overline{b}^T z, \\ \text{subject to} & A^T(y - z) = c, \\ & y \ge 0 \quad \text{and} \quad z \ge 0. \end{cases} \tag{2}$$

The general procedure for computing mathematically rigorous bounds is the following:

1. Assume we have bounds $\boldsymbol{x} = [\underline{x}, \overline{x}]$ on the optimal solution of the primal.
2. Use an LP solver to compute approximate Lagrange multipliers $\lambda = \lambda_{\underline{b}} - \lambda_{\overline{b}}$ for (2), where $\lambda_{\underline{b}} \approx y$ and $\lambda_{\overline{b}} \approx z$
3. Compute interval bounds on the dual residual $\boldsymbol{r} = [\underline{r}, \overline{r}]$, where the dual residual is $A^T\lambda - c$.
4. Take the lower bound on the solution to the primal of (2) to be

$$\mu = \inf(\lambda^T \boldsymbol{b} - \boldsymbol{r}^T \boldsymbol{x}) \qquad \text{(Formula (10) in [5])}, \tag{3}$$

For Formulation (1), we may use

$$c^T x \ge y^T b - \max\{r, 0\}^T \overline{x}, \tag{4}$$

in place of (3).

2 Notes of Caution

Here, we point out some errors implementers can make.

2.1 Misinterpretation of Results

The underlying idea is to obtain a bound on the optimum of a linear program (a lower bound if minimization and an upper bound if maximization) that is sharp if the solver returns a good approximation, but is a bound regardless of what the solver returns. Without care, this can easily lead to incorrect conclusions. Here, we outline errors implementers can easily make.

Example 2.1 *Consider*

$$\begin{array}{ll} \textit{maximize} & 2x_1 + 3x_2 \\ \textit{subject to} & x_1 + x_2 = 1, \\ & x_1 - x_2 = 2, \\ & 0 \leq x_1, x_2 \leq 2. \end{array}$$

A mathematically rigorous lower bound on the optimum can, in principle, be computed using Formula (16) that we present in §3.6. However, this particular problem is clearly infeasible, and, indeed, many solvers will return with an error flag set. Nonetheless, in such problems, a solver may return finite values for the Lagrange multipliers. Suppose such a solver returns $\lambda_{ub} = (0, -1700)^T$, $\lambda_{lb} = (-1700, 0)^T$, $\lambda_e = (0, -1700)^T$. *Applying (16) then gives an upper bound on an optimum of* $\nu = 0.004$.

This example illustrates that, in these computations, the upper bound ν that is computed is a **mathematically rigorous upper bound on the optimum only provided an optimum exists**. However, separate, related computations, as in [5, §4] can sometimes be used to prove a problem infeasible.

2.2 Pitfalls Associated with Particular Forms

In formulation (1), implicit upper bounds $\overline{x}$ are utilized to compute the lower bound μ on the primal optimum, while both implicit lower bounds $\underline{x}$ and $\overline{x}$ are utilized in formulation (2). However, such bounds must be interpreted carefully, as the following example shows.

Example 2.2 *Consider formulation (1) with* $c = (-1, 10^{-10})^T$, $A_e = (10^{-10}, 1)$ *and* $b_e = 1$.

Solving Example 2.2 using the Matlab release 2014a optimization toolbox function `linprog` with lower bounds 0 on x and no upper bounds gives $x \approx (10^{10}, 0)^T$, and an approximate dual variable is computed as $y \approx -10^{10}$. Using INTLAB [7, 8] Version 8, it is proven that $r < 0$, so the implicit upper bounds on x are irrelevant, that is, the point `linprog` returns is exactly feasible[1], and Formula (4) gives a correct lower bound (given here approximately) of about -10^{10}. However, if y is infeasible by a distance of, say 10^{-8}, and upper bounds of 2 or so are assumed for the components $\boldsymbol{x}$, an incorrect lower bound will be given. Moreover, the incorrect lower bound would still be close to the optimum, and

[1] This is understandable, since an interior point method was used.

the error may be undetected from examining results, until puzzling conclusions arise from, say, a branch and bound algorithm into which the computation is embedded.

Furthermore, in an interval branch and bound algorithm, typically we form a linear relaxation over a box defined by known bounds on x. If an interval branch and bound algorithm is to compute a lower bound on an objective over the feasible set in, say, $\boldsymbol{x} = ([0,2],[0,2]^T)$, the relaxation of the problem is of the form (1), and the box $\boldsymbol{x}$ is included only in computation of the lower bound, and not explicitly in the problem, a pessimistic lower bound may be computed, a bound may not be available because the problem without the bound constraints is unbounded, or, if y happens to be wildly infeasible, an incorrect lower bound on the objective over $\boldsymbol{x}$ could even be computed. Indeed, if we include the upper bound constraints $x_1 \leq 2$, $x_2 \leq 2$ explicitly in the formulation of the linear program in Example 2.2, `linprog` returns an optimum of approximately $c^T x \approx -2$, and a modification of Formula (4) gives a mathematically rigorous lower bound close to -2.

Our point here is that, **for mathematical rigor, guessed bounds on variables should not be heuristically provided when the original problem had none**.

3 A Collection of Formulas

As we illustrated in §2, the forms may not be appropriate for particular situations. Furthermore, the input forms corresponding to popular software packages for solution of linear programs, such as the Matlab optimization toolbox or common C++ or Fortran libraries, may not be in one of these two forms, requiring conversion to interface the output with the formulas; this can be an error-prone process. Indeed, the input forms in such packages and libraries may also be different from the internal representations used in the actual computations, and incorporating rigorous lower bound computation directly into the solver, using the internal representation directly, is probably the most accurate and least error-prone. It is for this reason we include a catalog of formulas for specific situations.

We use consistent notation throughout this section.

3.1 Notation

Throughout, n is the number of primal variables, m is the number of primal inequality constraints (if present), and m_e is the number of primal equality constraints (if present), so $c \in \mathbb{R}^n$, $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $A_\mathrm{e} \in \mathbb{R}^{m_\mathrm{e} \times n}$. We will call dual variables corresponding to primal inequality constraints $y \in \mathbb{R}^m$, dual variables corresponding to equality constraints $y_\mathrm{e} \in \mathbb{R}^{m_\mathrm{e}}$, dual variables corresponding to lower bound constraints $y_\mathrm{lb} \in \mathbb{R}^n$, and dual variables corresponding to upper bound constraints $y_\mathrm{ub} \in \mathbb{R}^n$; we denote corresponding approximate optimizing values returned by the solver for the dual variables by λ, λ_e, λ_lb,

and $\lambda_{\rm ub}$, and the nearest corresponding exactly optimizing dual variables λ^*, $\lambda^*_{\rm e}$, $\lambda^*_{\rm lb}$, and $\lambda^*_{\rm ub}$. We use simply x to denote an approximate primal optimizer returned by the solver (feasible or not) and x^* to denote corresponding exactly optimizing primal variables.

Throughout, we will use

$$v_+ = \max(v, 0) \quad \text{and} \quad v_- = \max(-v, 0), \tag{5}$$

where v is any quantity (subscripted or not), and where it is understood componentwise if v is a vector.

If the primal is posed as a minimization problem, we denote the mathematically rigorous lower bound on the primal optimum computed through our formulas by μ. If the primal is posted as a maximization problem, it makes sense to compute a mathematically rigorous upper bound on the optimum, which we denote by ν.

3.2 General Techniques

In writing down the dual problems, we have used the simple technique in [2], since it is general, easy to remember, and its use avoids errors.

In deriving the formulas for the general bounds, we use the general technique exemplified by [5, Formulas (5) through (9)]. Namely, it is assumed approximate values $\lambda_:$ (that is, λ, $\lambda_{\rm e}$, $\lambda_{\rm lb}$, and $\lambda_{\rm ub}$) for the dual variables have been returned, and the primal objective coefficients c are written in terms of the dual constraint residuals at $\lambda_:$. These constraint residuals are then rigorously bounded with either directed rounding or interval arithmetic, to obtain a lower bound on the primal objective values. In [5, Formula (9)], this was straightforward under implicitly assumed bounds on the primal solution x^*. Other primal formulations considered here, such as when the dual contains inequality constraints, can be handled with similar ease.

3.3 A General Form

We begin with the most complete formulation, corresponding to the most general call in the Matlab optimization toolbox, namely:

```
[x,fval,exitflag,output,lambda] = linprog(c,A,b,Aeq,beq,lb,ub)
```
(6)

$$\text{Primal:} \quad \begin{cases} \text{minimize} & c^T x, \\ \text{subject to} & Ax \le b, \\ & A_{\rm e} x = b_{\rm e}, \\ & \underline{x} \le x \le \overline{x}. \end{cases}$$

$$\text{Dual:} \quad \begin{cases} \text{maximize} & [b^T,\ \overline{x}^T,\ -\underline{x}^T,\ b_{\rm e}^T][y^T,\ y_{\rm ub}^T,\ y_{\rm lb}^T,\ y_{\rm e}^T]^T \\ \text{subject to} & A^T y + y_{\rm ub} - y_{\rm lb} + A_{\rm e}^T y_{\rm e} = c, \\ & y,\ y_{\rm ub},\ y_{\rm lb} \le 0. \end{cases} \tag{7}$$

Assuming an approximate dual solution $(\lambda, \lambda_{\rm lb}, \lambda_{\rm ub}, \lambda_{\rm e})$ and writing

$$c = A^T\lambda + \lambda_{\rm ub} - \lambda_{\rm lb} + A_{\rm e}^T\lambda_{\rm e} - r \tag{8}$$

where r is the dual constraint residual, we obtain

$$\begin{aligned} c^T x &= [A^T\lambda + \lambda_{\rm ub} - \lambda_{\rm lb} + A_{\rm e}^T\lambda_{\rm e} - r]^T x \\ &= [\lambda^T A + \lambda_{\rm ub}^T - \lambda_{\rm lb}^T + \lambda_{\rm e}^T A_{\rm e} - r^T]x \\ &= \lambda^T Ax + \lambda_{\rm ub}^T x - \lambda_{\rm lb}^T x + \lambda_{\rm e}^T A_{\rm e} x - r^T x \\ &\in \lambda^T \boldsymbol{b} + \lambda_{\rm ub}^T x - \lambda_{\rm lb}^T x + \lambda_{\rm e}^T b_{\rm e} - r^T x \\ &\subset \lambda^T \boldsymbol{b} + \lambda_{\rm e}^T b_{\rm e} + [\lambda_{\rm ub}^T - \lambda_{\rm lb}^T - r^T]\boldsymbol{x}, \end{aligned}$$

where $\boldsymbol{x} = [\underline{x}, \overline{x}]$ and $\boldsymbol{b} = [\inf(A\boldsymbol{x}),\ \min(\sup(A\boldsymbol{x}), b)]$. Therefore,

$$c^T x^* \geq \mu := \inf\left\{\lambda^T \boldsymbol{b} + \lambda_{\rm e}^T b_{\rm e} + [\lambda_{\rm ub}^T - \lambda_{\rm lb}^T - \boldsymbol{r}^T]\boldsymbol{x}\right\} \tag{9}$$

where $\boldsymbol{r}$ denotes a mathematically rigorous enclosure for r, such as can be computed with interval arithmetic applied to the point equation (8).

If λ, $\lambda_{\rm e}$, $\lambda_{\rm lb}$, **and** $\lambda_{\rm ub}$ **are feasible**, we may alternatively define μ as

$$\lambda^T \min\left\{\sup(A\boldsymbol{x}), b\right\} + \lambda_{\rm e}^T b_{\rm e} + \lambda_{\rm ub}^T \overline{x} - \lambda_{\rm lb}^T \underline{x} - \sup\{\boldsymbol{r}^T \boldsymbol{x}\},$$

Note that no "guesses" need to be made for this problem, since bounds $\boldsymbol{x}$ are explicitly given, and since the lower bound on b, although not given as part of the primal constraints, is a mathematically rigorous one. Also, note that, since the dual variables are constrained to be non-positive, the possibly pessimistic lower bound $\inf(A\boldsymbol{x})$ normally would have little effect on the lower bound for $c^T x^*$ if the dual variables are approximately feasible.

3.4 A Classical Standard Form

Primal problems of this form are commonly found in explanations of the basic simplex method (such as we do in [1, §9.4]), and may also be the internal form used in the computations in some simplex-method-based software. Furthermore, there are numerous elementary explanations on how to convert linear programs into this form.

$$\begin{aligned} &\text{Primal:} \quad \left\{ \begin{array}{ll} \text{maximize} & c^T x \\ \text{subject to} & A_{\rm e} x = b_{\rm e}, \\ & x \geq 0 \end{array} \right. \\ &\text{Dual:} \quad \left\{ \begin{array}{ll} \text{minimize} & b_{\rm e}^T y, \\ \text{subject to} & A_{\rm e}^T y \geq c, \end{array} \right. \end{aligned} \tag{10}$$

Note the usual convention in presentations of the basic simplex method is to pose the primal as a maximization problem. In the context of maximization, we compute an *upper* bound ν on the optimum objective value $c^T x^*$.

Suppose λ_{e} is an approximate optimizer of the dual, and

$$r = A^T \lambda_{\mathrm{e}} - c,$$

that is, $c = A^T \lambda_{\mathrm{e}} - r$. **In this case, we must assume** we have an implicit upper bound $\overline{x}$, such that $x \leq \overline{x}$ and $x^* \leq \overline{x}$, and we define $\boldsymbol{x} = [0, \overline{x}]$. We then have

$$\begin{array}{rcl} c^T x & = & (A^T \lambda_{\mathrm{e}} - r)^T x = \lambda_{\mathrm{e}}^T A x - r^T x = \lambda_{\mathrm{e}}^T b_e - r^T x \\ & \in & \lambda_{\mathrm{e}}^T b_e - r^T \boldsymbol{x}. \end{array} \tag{11}$$

Also,

$$\begin{array}{rcl} r^T \boldsymbol{x} & = & \displaystyle\sum_{i=1}^{n} r_{i,+}[0, \overline{x}_i] - \sum_{i=1}^{n} r_{i,-}[0, \overline{x}_i] \\ & \geq & \displaystyle -\sum_{i=1}^{n} r_{i,-} \overline{x}_i = -r_-^T \overline{x}. \end{array} \tag{12}$$

so (combining (11) and (12)) we have

$$c^T x \leq \nu := \lambda_{\mathrm{e}}^T b_e + r_-^T \overline{x}. \tag{13}$$

If λ_{e} is feasible, this reduces to simply $\lambda_{\mathrm{e}}^T b_e$.

Also note the bound depends in general on the quantity $\overline{x}$ that is not explicitly part of the problem. We thus recommend this form be used primarily when the solver returns feasible dual solutions λ_{e}, or when the λ_{e} returned by the solver can be easily adjusted to be feasible. In those cases, ν is particularly easy to compute as $\lambda_{\mathrm{e}}^T b_e$. If not, we encounter the pitfall of estimating $\overline{x}$.

3.5 Modified Classical Standard Form

If we replace "maximize $c^T x$" by "minimize $c^T x$" in (10), we obtain the formulation of [5, (2) and (3)], covered in [4].

3.6 Bounded Classical Standard Form

Here, we modify the classical standard form by inserting explicit bound constraints $\underline{x} \leq x \leq \overline{x}$. It is in this form that the problem is likely to occur in the

bounding process in branch and bound algorithms. We obtain

$$\text{Primal:} \quad \begin{cases} \text{maximize} & c^T x \\ \text{subject to} & A_{\mathrm{e}} x = b_{\mathrm{e}}, \\ & \underline{x} \le x \le \overline{x}. \end{cases}$$

$$\text{Dual:} \quad \begin{cases} \text{minimize} & (\overline{x}^T, -\underline{x}^T, b_{\mathrm{e}}^T) \begin{pmatrix} y_{\mathrm{ub}} \\ y_{\mathrm{lb}} \\ y_{\mathrm{e}} \end{pmatrix}, \\ \text{subject to} & (I, -I, A_{\mathrm{e}}^T) \begin{pmatrix} y_{\mathrm{ub}} \\ y_{\mathrm{lb}} \\ y_{\mathrm{e}} \end{pmatrix} = c, \\ & y_{\mathrm{ub}}, y_{\mathrm{lb}} \le 0, \end{cases} \tag{14}$$

where I is the n by n identity matrix. Note that, besides adding an upper bound $\overline{x}$, we have slightly generalized the classical standard form by allowing $\underline{x}$ to be an arbitrary value, possibly less than 0. Note also that this form can be easily transformed into the classical standard form by well-publicized techniques.

Suppose $(\lambda_{\mathrm{ub}}^T, \lambda_{\mathrm{lb}}^T, \lambda_{\mathrm{e}}^T)^T$ is an approximate dual optimizer, and define the dual constraint residual r

$$r = (I, -I, A_{\mathrm{e}}^T) \begin{pmatrix} \lambda_{\mathrm{ub}} \\ \lambda_{\mathrm{lb}} \\ \lambda_{\mathrm{e}} \end{pmatrix} - c.$$

Solving this equation for c and substituting into $c^T x$ gives

$$\begin{aligned} c^T x &= [(\lambda_{\mathrm{ub}}, -\lambda_{\mathrm{lb}}, A^T \lambda_{\mathrm{e}}) - r]^T x \\ &= \lambda_{\mathrm{ub}}^T x - \lambda_{\mathrm{lb}}^T x + \lambda_{\mathrm{e}}^T A x - r^T x \\ &= \lambda_{\mathrm{ub}}^T x - \lambda_{\mathrm{lb}}^T x + \lambda_{\mathrm{e}}^T b_{\mathrm{e}} - r^T x \\ &= \lambda_{\mathrm{e}}^T b_{\mathrm{e}} + (\lambda_{\mathrm{ub}}^T - \lambda_{\mathrm{lb}}^T - r^T) x \\ &\in \lambda_{\mathrm{e}}^T b_{\mathrm{e}} + (\lambda_{\mathrm{ub}}^T - \lambda_{\mathrm{lb}}^T - \boldsymbol{r}^T) \boldsymbol{x}. \end{aligned} \tag{15}$$

This gives us

$$c^T x \le \nu := \sup(\lambda_{\mathrm{e}}^T b_{\mathrm{e}} + (\lambda_{\mathrm{ub}}^T - \lambda_{\mathrm{lb}}^T - \boldsymbol{r}^T) \boldsymbol{x}) \tag{16}$$

where $\boldsymbol{r}$ is a mathematically rigorous enclosure for r and $\boldsymbol{x} = [\underline{x}, \overline{x}]$.

3.7 Simple Form

This form corresponds to the simplest way the Matlab[2] function `linprog` can be invoked, namely,

```
x = linprog(c,A,b)
```
(17)

[2]and also Octave [6, 3]

$$\text{Primal:} \quad \begin{cases} \text{minimize} & c^T x \\ \text{subject to} & Ax \le b, \end{cases}$$

$$\text{Dual:} \quad \begin{cases} \text{maximize} & b^T y \\ \text{subject to} & A^T y = c, \\ & y \le 0. \end{cases} \tag{18}$$

Suppose λ is an approximate solution of the dual, so the dual constraint residual at λ is

$$r = A^T \lambda - c. \tag{19}$$

If we proceed as in the previous lower bound derivations, we obtain $c^T x = (A^T y - r)^T x$, and bounding this expression requires a priori bounds on the primal variables x. Since the signs of the components of $A^T y - r$ are generally not known in this case, both lower bounds and upper bounds $\underline{x}$ and $\overline{x}$ on the optimal primal solution x^* to (18) need to be known a priori, a pitfall. **If such bounds are known**, define $\boldsymbol{x} = [\underline{x}, \overline{x}]$, to obtain

$$\begin{aligned} c^T x &= (A^T y - r)^T x \\ &\in (A^T y - r)^T \boldsymbol{x} \\ &\ge \mu := \inf((A^T y - r)^T \boldsymbol{x}). \end{aligned} \tag{20}$$

3.8 Bounded Simple Form

Here, the form (18) is modified by adding explicit bound constraints on the primal variables x. We have

```
x = linprog(c,A,b,[],[],lb,ub)
```
(21)

$$\text{Primal:} \quad \begin{cases} \text{minimize} & c^T x \\ \text{subject to} & Ax \le b, \\ & \underline{x} \le x \le \overline{x} \end{cases}$$

$$\text{Dual:} \quad \begin{cases} \text{maximize} & \left(b^T, \overline{x}^T, -\underline{x}^T\right) \begin{pmatrix} y \\ y_{\text{ub}} \\ y_{\text{lb}} \end{pmatrix} \\ \text{subject to} & \left(A^T, I, -I\right) \begin{pmatrix} y \\ y_{\text{ub}} \\ y_{\text{lb}} \end{pmatrix} = c, \\ & y \le 0, y_{\text{ub}} \le 0, y_{\text{lb}} \le 0, \end{cases} \tag{22}$$

where I is the n by n identity matrix.

Proceeding as in the previous sections, the dual constraint residual is

$$r = A^T \lambda + \lambda_{\text{ub}} - \lambda_{\text{lb}} - c, \tag{23}$$

and, setting $\boldsymbol{x} = [\underline{x}, \overline{x}]$, we may bound $c^T x$ by

$$\begin{aligned} c^T x &= (A^T\lambda + \lambda_{\mathrm{ub}} - \lambda_{\mathrm{lb}} - r)^T x && (24)\\ &\in (A^T\lambda + \lambda_{\mathrm{ub}} - \lambda_{\mathrm{lb}} - r)^T \boldsymbol{x} && (25)\\ &\geq \mu := \inf\left[(A^T\lambda + \lambda_{\mathrm{ub}} - \lambda_{\mathrm{lb}} - r)^T \boldsymbol{x}\right]. && (26) \end{aligned}$$

3.9 General Form Without Bound Constraints

This corresponds to e.g. the form

```
x = linprog(c,A,b,Aeq,beq)
```

in the Matlab optimization toolbox. We have

$$\begin{aligned} &\text{Primal:} \quad \begin{cases} \text{minimize} & c^T x, \\ \text{subject to} & Ax \leq b, \\ & A_{\mathrm{e}} x = b_{\mathrm{e}}, \end{cases} \\ &\text{Dual:} \quad \begin{cases} \text{maximize} & b^T y + b_{\mathrm{e}}^T y_{\mathrm{e}}, \\ \text{subject to} & A^T y + A_{\mathrm{e}}^T y_{\mathrm{e}} = c, \\ & y \leq 0 \end{cases} \end{aligned} \qquad (27)$$

In the primal, no bounds on x are assumed. Proceeding as in [5] and the previous sections of this work, the dual constraint residual is

$$r = (A^T, A_{\mathrm{e}}^T) \begin{pmatrix} \lambda \\ \lambda_{\mathrm{e}} \end{pmatrix} - c,$$

so

$$\begin{aligned} c^T x &= \left((A^T, A_{\mathrm{e}}^T) \begin{pmatrix} \lambda \\ \lambda_{\mathrm{e}} \end{pmatrix} - r\right)^T x && (28)\\ &= \lambda^T A x + \lambda_{\mathrm{e}}^T b_e - r^T x \\ &\geq \mu := \inf\left\{(\lambda^T A - r^T)\boldsymbol{x}\right\} + \lambda_{\mathrm{e}}^T b_e, \end{aligned}$$

where $\boldsymbol{x} = [\underline{x}, \overline{x}]$ represents independently obtained lower and upper bounds on the optimal solution x^*. However, the bounds $\underline{x}$ and $\overline{x}$ are often not available. For example, consider the LP

$$\begin{cases} \text{minimize} & -x_1 - 2x_2 \\ \text{subject to} & \begin{pmatrix} 1 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leq \begin{pmatrix} 3 \\ 3 \end{pmatrix}, \\ & \begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \end{cases} \qquad (29)$$

The minimum to (29) is 0, but the set of optimizing points is an unbounded ray; a lower bound on the first coordinate and an upper bound on the second coordinate of x are not obtainable, and any lower bound μ obtained from Formula (27) could be incorrect or pessimistic.

4 Summary and Recommendations

We have examined pitfalls of the formulas for mathematically rigorous bounds on optima of linear programs proposed in [5], and have presented various related alternative formulas. Which formula is employed depends on which LP solver is being used and the context in which it is accessed. For example, if the LP solver is being accessed from Matlab's programming interface, one of the formulas in §3.3 or §3.8 is appropriate, while use of the forms in §3.7 or §3.9 could lead to incorrect results unless rigorous a priori bounds on the optimal solution of the corresponding LP are known a priori.

When the classical simplex method is used and it is possible to modify the code to embed computation of a rigorous bound, it may be more problematical, since one is apt to find the form in §3.4 within the code, but an upper bound $\overline{x}$ is not available. In such instances, it would be preferable to approach the task from a higher level, if the original problem actually had bound constraints, as in §3.6.

Fortunately, the forms with bounds, that is, the forms in §3.3, §3.6, and §3.8 are the ones most likely to be encountered in solving linear relaxations in branch and bound algorithms for mixed integer linear programs and global non-convex optimization.

Other forms can be derived as needed, according to the basic principles from [5] and those illustrated in this work.

References

[1] Azmy S. Ackleh, Edward James Allen, Ralph Baker Kearfott, and Padmanabhan Seshaiyer. *Classical and Modern Numerical Analysis, Theory, Methods and Practice*. Chapman & Hall/CRC Numerical Analysis and Scientific Computing. CRC Press, Boca Raton, Florida, 2010.

[2] Arthur T. Benjamin. Sensible rules for remembering duals—the S-O-B method. *SIAM Rev.*, 37(1):85–87, 1995.

[3] John W. Eaton, David Bateman, and S**o**ren Hauberg. *GNU Octave Version 3.0.1 Manual: A High-Level Interactive Language for Numerical Computations*. CreateSpace Independent Publishing Platform, 2009. ISBN 1441413006.

[4] Jared T. Guilbeau, Md. Istiaq Hossain, Sam D. Karhbet, Ralph Baker Kearfott, Temitope S. Sanusi, and Lihong Zhao. A review of computation of mathematically rigorous bounds on optima of linear programs. *Journal of Global Optimization*, to appear, 2016.

[5] Arnold Neumaier and Oleg Shcherbina. Safe bounds in linear and mixed-integer programming. *Math. Prog.*, 99(2):283–296, March 2004.

[6] Octave community. GNU Octave 3.8.1, 2014. `www.gnu.org/software/octave/`.

[7] Siegfried M. Rump. INTLAB–INTerval LABoratory. In Tibor Csendes, editor, *Developments in Reliable Computing: Papers presented at the International Symposium on Scientific Computing, Computer Arithmetic, and Validated Numerics, SCAN-98, in Szeged, Hungary*, volume 5(3) of *Reliable Computing*, pages 77–104, Dordrecht, Netherlands, 1999. Kluwer Academic Publishers. URL: `http://www.ti3.tu-harburg.de/rump/intlab/`.

[8] Siegfried M. Rump. INTLAB – INTerval LABoratory, 1999–2015. `http://www.ti3.tu-harburg.de/rump/intlab/`.