J Glob Optim (2009) 45:645–666
DOI 10.1007/s10898-009-9403-5

On optimal zero-preserving corrections for inconsistent linear systems

Paula Amaral · Luís M. Fernandes · Joaquim Júdice · Hanif D. Sherali

Received: 20 April 2007 / Accepted: 19 January 2009 / Published online: 11 February 2009

Abstract This paper addresses the problem of finding an optimal correction of an inconsistent linear system, where only the nonzero coefficients of the constraint matrix are allowed to be perturbed for reconstructing a consistent system. Using the Frobenius norm as a measure of the distance to feasibility, a nonconvex minimization problem is formulated, whose objective function is a sum of fractional functions. A branch-and-bound algorithm for solving this nonconvex program is proposed, based on suitably overestimating the denominator function for computing lower bounds. Computational experience is presented to demonstrate the efficacy of this approach.

Keywords Infeasible systems · Nonlinear programming · Fractional programming · Global optimization

1 Introduction

Mathematical modeling of real-life environments, which are often complex and ambiguous, is frequently the first step in the process of deriving insights into making the best decisions. In this paper, we focus on mathematical models described by linear systems. Translating

P. Amaral (✉)
Departamento de Matemática, Universidade Nova de Lisboa, Monte da Caparica, Portugal
e-mail: paca@fct.unl.pt

L. M. Fernandes
Instituto Politécnico de Tomar and Instituto Telecomunicações, Coimbra, Portugal

J. Júdice
Departamento de Matemática, Universidade de Coimbra and Instituto Telecomunicações, Coimbra, Portugal

H. D. Sherali
Grado Department of Industrial & Systems Engineering, Virginia Polytechnic Institute & State University, Blacksburg, VA, USA

a decision problem from a natural language into a mathematical structure, as is done in linear programming or constraint programming, is a powerful tool to understand the problem, construct a suitable abstraction, find appropriate solutions, and conduct sensitivity analysis. In general, modeling such problems is usually a compromise between accuracy and tractability. Frequently, deterministic data is assumed even when we are dealing with estimates alone. The difficulty of solving a multiparametric problem is a handicap for dealing with multiple and independent perturbations on data. On the other hand, methodologies that can incorporate inherent sources of uncertainty in the mathematical model itself are better suited. Mathematical models can also be restructured from the viewpoint of feasibility for a posterior implementation. In this case, as the result of an overly demanding approach, it can happen that the mathematical model is inconsistent, in the sense that there are no feasible solutions for the model. As the model evidently reflects the purposes of the decision-maker, some minimally corrective action must therefore be taken in order to overcome the deadlock created by the inconsistency. As discussed in [11,14,20–25,34], it is possible to analyze the model to extract some relevant information concerning the infeasibility of the system.

It may also be interesting to evaluate the distance to feasibility for ill-posed problems [32,33], or to study some theoretical aspects regarding duality concepts for inconsistent systems [18,38,39]. Additionally, from a practical point of view, it is worthwhile to perturb the model in order to find a feasible neighboring formulation in a defined sense [4–7].

Removing constraints is a possible action to repair the model. In [26], the authors proposed a method to remove constraints in order to obtain a feasible set, based on an hierarchical classification of constraints. Another possibility is to remove a minimal set of constraints. This problem is however NP-hard [11]. Heuristic approaches can be implemented based on the generation of Irreducible Inconsistent Systems (IISs) [14,15,37]. An IIS is a set of inconsistent constraints for which every proper subsystem is consistent. There exist several algorithms for finding an IIS [1,2,13,30,42]. Most commercial solvers, such as CPLEX [17] and LINDO [29], have efficient implementations for finding IISs. A feasible set of constraints can be obtained by iteratively generating an IIS and removing one constraint from it, and then repeating this process. In the end it is sometimes possible to reintroduce some constraints that no longer need to be removed to achieve feasibility. This type of approach ignores some inherent relations between the decision variables, which might be acceptable in certain cases, as, for example, when the deleted restrictions are soft constraints. However, this procedure can be completely inadequate in situations where it is preferable to derive a feasible model that essentially retains the parent constraints, but due to parameters approximation, it is admissible to perturb those parameters to some extent.

Consider a general inconsistent linear system:

$$\sum_{j=1}^{n} a_{ij}x_j \leq b_i, \quad i = 1, \ldots, m_0,$$

$$\sum_{j=1}^{n} a_{ij}x_j = b_i, \quad i = m_0 + 1, \ldots, m,$$

defined by the matrix $A = [a_{ij}] \in \mathbb{R}^{m\times n}$ and right-hand side (RHS) $b \in \mathbb{R}^m$. Note that this inconsistent system might typically be a manageably sized IIS arising from a larger parent model. We are interested in finding an optimal perturbation of the RHS, $p \in \mathbb{R}^m$, and the matrix of coefficients, $H = [h_{ij}] \in \mathbb{R}^{m\times n}$, which minimizes a measure of the total perturbation, given by the function $\varphi(H, p)$. Hence, we seek to solve the following nonlinear program:

$$\begin{aligned} &\text{Minimize} \quad \varphi(H, p) \\ &\text{subject to} \quad \sum_{j=1}^{n}(a_{ij} + h_{ij})x_j \le b_i + p_i, \quad i = 1, \ldots, m_0, \\ &\qquad \sum_{j=1}^{n}(a_{ij} + h_{ij})x_j = b_i + p_i, \quad i = m_0 + 1, \ldots, m, \\ &\qquad x \in \Omega, \end{aligned} \tag{1}$$

where Ω is a convex set containing all the constraints that are not to be perturbed. The most common choice for this set (as assumed herein) is $\Omega = \{x : l \le x \le u\}$, where $-\infty < l_j < u_j < \infty$, $\forall j = 1, \ldots, n$. The inclusion of these bounds are assumed to be logically, technically, and practically appropriate and therefore, not subject to perturbations. However, whenever such bounds are open to perturbation, we can incorporate the relevant bounding relationships within the structural constraints in (1), and assume that the decision-maker has subsequently imposed hard logical bounds within Ω. These bounds are not only required for technical algorithmic reasons, but also, from a practical point of view, they represent an effective domain of search for the user. It is also perfectly reasonable to assume that the underlying application can reasonably prompt an acceptable decision variable interval. Many practical applications logically imply the nonnegativity of variables, thus providing a natural lower bound, that cannot be reduced. A sufficiently large number can be used as an upper bound.

Regarding previous methods on this subject, to our knowledge, Vatolin [40] was the first to present an algorithm for finding an optimal correction to an inconsistent linear system. In his work, a family of objective functions (to minimize the distance between the original and the corrected problem) was considered, and a set of linear programming problems had to be solved in order to find an optimal correction. The familiar norms $\|\cdot\|_\infty$ and generalized norms $\|\cdot\|_{\ell_1}$ and $\|\cdot\|_{\ell_\infty}$ for matrices [1] are particular cases of the proposed family of functions. It was further shown in [3] for the particular case, where $m_0 = m$ and $\Omega = \mathbb{R}^n$, that $2n + 1$ linear programs are required to be solved for the l_1 and ∞ norms, and 2^n for the l_∞ norm. Moreover, a fixed structure for the data perturbation always results. More precisely, the optimal solution for (1) consists of changes in only one column of (A, b) for norms l_1 or ∞, while for the l_∞ norm, the perturbation of the coefficients of every row is the same, differing only in sign. Therefore, instead of random small perturbations, a constant pattern for the correction matrix is obtained when the proximity criteria between the original and the perturbed model is measured by these norms.

This result provided a strong motivation in [8] for investigating the effect on the perturbation matrix by using the Frobenius norm, that is, by setting $\varphi(H, p) = \frac{1}{2} \| [H, p] \|_F^2$ in Program (1), where for $G = [g_{ij}]$, we have $\| G \|_F^2 = \sum_i \sum_j g_{ij}^2$. A branch-and-bound algorithm based on the so-called reformulation-linearization-convexification technique (RLT) [35] was proposed for finding an optimal correction [8]. Computational experience reported in [8] showed that this approach was successful for handling medium-scale problems. Furthermore, the importance of finding a model correction approach that preserves the original zeros was stressed. In fact, a zero in a constraint means that the corresponding variable does not contribute to it. So, typically in practice, the correction of the linear system into a feasible one should not modify the zeros of the matrix of the constraints of the problem.

[1] For $G = [g_{ij}]$, we have $\| G \|_\infty = \max_i \sum_j |g_{ij}|$, $\| G \|_{\ell_1} = \sum_{ij} |g_{ij}|$, $\| G \|_{\ell_\infty} = \max_{ij} |g_{ij}|$.

In this paper, we address the case where no perturbations are allowed for the zero coefficients of the constraint matrix for finding an optimal correction using the Frobenius norm, which leads to a different formulation than the one presented in [8]. The resulting problem turns out to be a nonconvex program with an objective function that is a sum of fractional functions, subject to a set of linear constraints.

When applied to a nonconvex programming problem, existing unconstrained optimization techniques can guarantee at most a local optimum [9,27]. A certificate is required to confirm that such a solution found by an algorithm is a global optimum for the problem under consideration. The computation of lower bounds is the most common way for achieving such a certificate and has motivated and justified the development of a branch-and-bound algorithm for solving this nonconvex program.

The procedure we adopt to compute lower bounds shares some common features with the RLT approach and relies on overestimating the denominator function. Computational experience with medium-scale problems show the appropriateness of the proposed algorithm for solving this problem. Numerical results included in this paper also indicate that, on the other hand, a local solver was unable to find a global optimum for almost half of the instances tested by the branch-and-bound algorithm.

The remainder of this paper is organized as follows. The formulation of the optimal correction problem as a nonconvex fractional program is discussed in Sect. 2. The branch-and-bound algorithm is introduced in Sect. 3. Some computational experience is reported in Sect. 4, and 5 provides concluding remarks.

2 Fractional programming formulation

Consider the optimal correction problem (1) with the Frobenius norm:

$$(P^0): \ \min \varphi = \frac{1}{2}\left(\sum_{i=1}^{m}\sum_{j=1}^{n} h_{ij}^2 + \sum_{i=1}^{m} p_i^2\right)$$

subject to

$$\begin{aligned} &\sum_{j=1}^{n}(a_{ij}+h_{ij})x_j \le b_i + p_i, \quad i = 1,\dots m_0 \\ &\sum_{j=1}^{n}(a_{ij}+h_{ij})x_j = b_i + p_i, \quad i = m_0+1,\dots,m \\ &x \in \Omega. \end{aligned} \tag{2}$$

Let

$$I = \{(i,j) : 1 \le i \le m, 1 \le j \le n, \quad a_{ij} \ne 0\} \tag{3}$$

$$\overline{I} = \{(i,j) : 1 \le i \le m, 1 \le j \le n, \quad a_{ij} = 0\} \tag{4}$$

$$I_i = \{j : 1 \le j \le n, (i,j) \in I\}, \quad i = 1,\dots,m. \tag{5}$$

By assuming that $h_{ij} = 0$ for all $(i,j) \in \overline{I}$, we obtain the following formulation of the zero-preserving optimal correction problem:

$$(P^1): \ \min \varphi = \frac{1}{2}\left(\sum_{(i,j)\in I} h_{ij}^2 + \sum_{i=1}^{m} p_i^2\right) \tag{6}$$

subject to

$$\sum_{j\in I_i}(a_{ij}+h_{ij})x_j \le b_i + p_i, \quad i = 1,\dots m_0, \tag{7}$$

$$\sum_{j\in I_i}(a_{ij}+h_{ij})x_j = b_i + p_i, \quad i = m_0+1,\dots m, \tag{8}$$

$$x \in \Omega. \tag{9}$$

The Lagrangian function of P^1 is

$$L(x,h,p,\lambda) = \min \frac{1}{2}\left(\sum_{(i,j)\in I} h_{ij}^2 + \sum_{i=1}^{m} p_i^2\right) + \sum_{i=1}^{m}\lambda_i\left(\sum_{j\in I_i}(a_{ij}+h_{ij})x_j - b_i - p_i\right), \tag{10}$$

with $\lambda_i \ge 0$ for $i = 1,\dots,m_0$. As in [8], we attempt to find an equivalent representation of problem P^1 in only the x-variables. This is done by eliminating the unrestricted variables h_{ij}, $(i,j) \in I$ and $p_i, i = 1,\dots,m$, using the necessary and sufficient KKT conditions for Problem P^1 whenever $x \in \Omega$ is fixed (whence it is a linearly constrained convex quadratic program). It follows from these conditions that $\lambda_i \ge 0, i = 1,\dots,m_0$, and

$$\frac{\partial L}{\partial h_{ij}} = 0 \Leftrightarrow h_{ij} + \lambda_i x_j = 0 \Leftrightarrow h_{ij} = -\lambda_i x_j, \quad (i,j) \in I, \tag{11}$$

$$\frac{\partial L}{\partial p_i} = 0 \Leftrightarrow p_i - \lambda_i = 0 \Leftrightarrow p_i = \lambda_i, \quad i = 1,\dots,m, \tag{12}$$

$$\lambda_i\left(\sum_{j\in I_i}(a_{ij}+h_{ij})x_j - b_i - p_i\right) = 0, \quad i = 1,\dots,m_0. \tag{13}$$

Using (11) and (12) to eliminate the variables h_{ij} and p_i, we can write (13) in the form:

$$\lambda_i\left(\sum_{j\in I_i} a_{ij}x_j - b_i - \lambda_i\left(\sum_{j\in I_i} x_j^2 + 1\right)\right) = 0, \quad i = 1,\dots,m_0. \tag{14}$$

From (14) we have,

$$\lambda_i = 0 \text{ or } \lambda_i = \frac{\sum_{j\in I_i} a_{ij}x_j - b_i}{\sum_{j\in I_i} x_j^2 + 1}, \quad i = 1,\dots,m_0. \tag{15}$$

Since $\lambda_i \ge 0$, for all $i = 1,\dots,m_0$, we select

$$\lambda_i = \frac{\left(\sum_{j\in I_i} a_{ij}x_j - b_i\right)^+}{\sum_{j\in I_i} x_j^2 + 1}, \quad i = 1,\dots,m_0, \tag{16}$$

where $(z)^+ = \max\{0, z\}$.

Furthermore, repeating the above simplification in (13–15), for the equality constraints (8), we get

$$\lambda_i = \frac{\left(\sum_{j\in I_i} a_{ij}x_j - b_i\right)}{\sum_{j\in I_i} x_j^2 + 1}, \quad i = m_0+1,\dots,m. \tag{17}$$

Due to the definition of λ_i given by (16) and (17), and using (11) and (12), the constraints (7) and (8) automatically hold. Hence, this produces a KKT solution, and therefore an optimum, to the restricted problem with x fixed. Also, by (11) and (12), the objective function in (6) is then given by $(1/2)\sum_{i=1}^{m}\lambda_i^2\left[1+\sum_{j\in I_i}x_j^2\right]$. Therefore, Problem P^1 defined by (6–9) is equivalent to the following nonconvex, nondifferentiable program in the x-variables:

$$(P^2):\ \min_{x\in\Omega}\widetilde{\varphi}(x)=\frac{1}{2}\left(\sum_{i=1}^{m_0}\left[\frac{\left(\sum_{j\in I_i}a_{ij}x_j-b_i\right)^{+2}}{\sum_{j\in I_i}x_j^2+1}\right]+\sum_{i=m_0+1}^{m}\left[\frac{\left(\sum_{j\in I_i}a_{ij}x_j-b_i\right)^{2}}{\sum_{j\in I_i}x_j^2+1}\right]\right). \tag{18}$$

By solving P^2 globally and determining an optimal x-solution, we can obtain a corresponding set of global optimal values of h_{ij} and p_i by first evaluating $\lambda_i, i=1,\ldots,m$, using (16) and (17) and then using the formulae (11) and (12).

To illustrate this formulation, consider the linear system:

$$\begin{aligned} x_1+2x_2 &\le 6\\ x_1+x_3 &\le -7\\ 2x_1+x_2+x_3 &\le -5\\ x_1+2x_3 &= 3\\ (x_1,x_2,x_3) &\in \Omega. \end{aligned}$$

The formulation of the zero preserving problem leads to the following nonconvex and nondifferentiable program:

$$\min_{x\in\Omega}\widetilde{\varphi}(x)=\frac{1}{2}\left(\frac{(x_1+2x_2-6)^{+2}}{1+x_1^2+x_2^2}+\frac{(x_1+x_3+7)^{+2}}{1+x_1^2+x_3^2}+\frac{(2x_1+x_2+x_3+5)^{+2}}{1+x_1^2+x_2^2+x_3^2}+\frac{(x_1+2x_3-3)^{2}}{1+x_1^2+x_3^2}\right).$$

3 Branch–and–Bound algorithm based on overestimating the denominator of the objective function

As discussed in the previous section, the objective function of Problem P^2 in (18) is nondifferentiable, beside being nonconvex. However, by introducing additional nonnegative variables $v_i, i=1,\ldots,m_0$, we obtain the following equivalent differentiable representation of Problem P^2:

$$(P^3): \min f(x,v) = \sum_{i=1}^{m_0} \frac{v_i^2}{1+\sum_{j\in I_i} x_j^2} + \sum_{i=m_0+1}^{m} \frac{\left(\sum_{j\in I_i} a_{ij}x_j - b_i\right)^2}{\sum_{j\in I_i} x_j^2 + 1} \tag{19}$$

s.t.

$$v_i \geq \sum_{j\in I_i} a_{ij}x_j - b_i, \quad i = 1,\ldots,m_0 \tag{20}$$

$$x \in \Omega \tag{21}$$

$$v_i \geq 0, \quad i = 1,\ldots,m_0. \tag{22}$$

The equivalence of P^2 and P^3 comes from the fact that we have a minimization problem, where v_i, for $i = 1,\ldots,m_0$, appears in the numerator of the objective function and $v_i \geq 0$.

In this section we introduce a branch–and–bound algorithm for finding a global optimal solution to this problem P^3. The use of branch–and–bound methods is widely used for continuous nonconvex problems [19,26,28,35,36]. The proposed branch–and–bound procedure constructs a binary tree based on the partitioning of the hyperrectangle $\Omega \equiv [l,u]$ into subhyperrectangles $\Omega^k \equiv [l^k,u^k]$, where $k = 1,\ldots,K$ indexes the nodes that are generated at any stage. Here, the node with label k, for $1 \leq k \leq K$, is associated with the nonlinear program:

$$(P_k^3): \min f(x,v) = \sum_{i=1}^{m_0} \frac{v_i^2}{1+\sum_{j\in I_i} x_j^2} + \sum_{i=m_0+1}^{m} \frac{\left(\sum_{j\in I_i} a_{ij}x_j - b_i\right)^2}{1+\sum_{j\in I_i} x_j^2} \tag{23}$$

s.t.

$$v_i \geq \sum_{j\in I_i} a_{ij}x_j - b_i, \quad i = 1,\ldots,m_0, \tag{24}$$

$$l^k \leq x \leq u^k \tag{25}$$

$$v_i \geq 0, \quad i = 1,\ldots,m_0. \tag{26}$$

The proposed branch–and–bound algorithm uses the following techniques for obtaining lower and upper bounds for the optimal value of Problem P_k^3. The procedure for obtaining a lower bound for P_k^3 is based on the linearization of the functions in each of the denominators of the objective function terms. To our knowledge this technique was first proposed in [19], but also, the bound-factor product of the RLT scheme in [35] given by $[(x_j^k - l_j)(u_j^k - x_j)]_L \geq 0$ produces the same inequality, as discussed at the end of this section. More specifically, we replace x_j^2 in each term of the denominator $1+\sum_{j\in I_i} x_j^2$ of the objective function (23), by the affine function $\delta_j^k x_j + \beta_j^k$, such that:

$$\delta_j^k x_j + \beta_j^k \geq x_j^2 \text{ for } x_j \in [l_j^k, u_j^k], \tag{27}$$

where this function yields the concave envelope of x_j^2 over the interval $[l_j^k, u_j^k]$, as depicted in Fig. 1. Hence,

$$\delta_j^k = \frac{(u_j^k)^2 - (l_j^k)^2}{u_j^k - l_j^k} = u_j^k + l_j^k,$$
$$\beta_j^k = -u_j^k l_j^k.$$

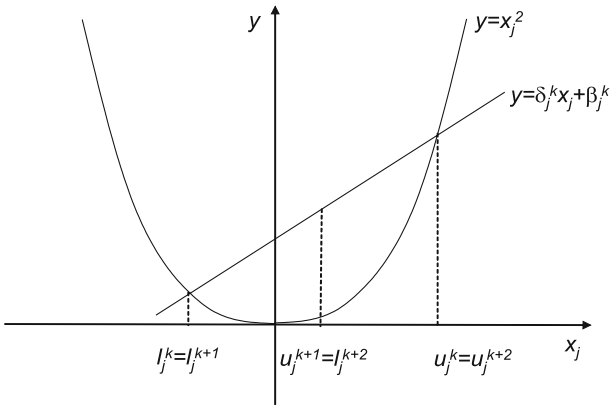

Fig. 1 Linear approximation of x_j^2

This yields the following lower bounding problem:

$$(LB(P_k^3)) : \min g^k(x,v) = \sum_{i=1}^{m_0} \frac{v_i^2}{1+\sum_{j\in I_i}\delta_j^k x_j+\beta_j^k} + \sum_{i=m_0+1}^{m} \frac{\left(\sum_{j\in I_i} a_{ij}x_j - b_i\right)^2}{1+\sum_{j\in I_i}\delta_j^k x_j+\beta_j^k}$$

s.t.

$$v_i \geq \sum_{j\in I_i} a_{ij}x_j - b_i, \quad i=1,\ldots,m_0,$$
$$l^k \leq x \leq u^k$$
$$v_i \geq 0, \quad i=1,\ldots,m_0.$$

The following result holds, where henceforth, $v(Pr)$ denotes the optimal value of any problem Pr.

Theorem 1 *Let* (x^{*k}, v^{*k}) *be an optimal solution of Problem* $LB(P_k^3)$. *Then* $g^k(x^{*k}, v^{*k}) \leq v(P_k^3)$ *and* $v(P^3) \leq f(x^{*k}, v^{*k})$.

Proof Let Q and Q^k be, respectively, the set of feasible solutions of P^3 and $LB(P_k^3)$. Q^k is also the set of feasible solutions of P_k^3. Then $Q^k \subseteq Q$. By the construction (27), we have,

$$\sum_{j\in I_i}(\delta_j^k x_j+\beta_j^k) \geq \sum_{j\in I_i} x_j^2 \quad \text{for} \quad x_j \in [l_j^k, u_j^k]. \tag{28}$$

Therefore,

$$\frac{1}{1+\sum_{j\in I_i}(\delta_j^k x_j+\beta_j^k)} \leq \frac{1}{1+\sum_{j\in I_i} x_j^2} \quad \text{for} \quad x_j \in [l_j^k, u_j^k]$$

and $g^k(x,v) \leq f(x,v)$ for all $(x,v) \in Q^k$. Hence,

$$\min\left\{g^k(x,v) : (x,v)\in Q^k\right\} \leq \min\left\{f(x,v) : (x,v)\in Q^k\right\}.$$

Since $Q^k \subseteq Q$, then $v(P^3) \leq f(x^{*k}, v^{*k})$. □

Next, we show that $LB(P_k^3)$ is a convex program, by proving that its objective function is convex over the constraint set of the program. Toward this end, let $r_i^k \equiv 1 - \sum_{j\in I_i} u_j^k l_j^k$ and write $LB(P_k^3)$ in the following equivalent form:

$$\min \widehat{g}^k(x,v) = \sum_{i=1}^{m} \frac{v_i^2}{r_i^k + \sum_{j\in I_i}(u_j^k + l_j^k)x_j} \tag{29}$$

$$s.t.$$

$$v_i \geq \sum_{j\in I_i} a_{ij}x_j - b_i, \quad i = 1,\ldots,m_0, \tag{30}$$

$$v_i = \sum_{j\in I_i} a_{ij}x_j - b_i, \quad i = m_0+1,\ldots,m, \tag{31}$$

$$l^k \leq x \leq u^k \tag{32}$$

$$v_i \geq 0, \quad i = 1,\ldots,m_0.$$

We start by proving the following technical result.

Theorem 2 *The function*

$$\zeta_i(v_i, x_1, \ldots, x_{n_i}) \equiv \frac{v_i^2}{c_i + \sum_{j=1}^{n_i} d_j x_j} \tag{33}$$

is convex on $\mathcal{Z} \equiv \mathbb{R} \times \{(x_1,\ldots,x_{n_i}) \in \mathbb{R}^{n_i} : c_i + \sum_{j=1}^{n_i} d_j x_j > 0\}$.

Proof The gradient and hessian of ζ_i are given by

$$\nabla\zeta_i = \begin{bmatrix} \frac{2v_i}{c_i+\sum_{j=1}^{n_i} d_j x_j} \\ \frac{-d_1 v_i^2}{(c_i+\sum_{j=1}^{n_i} d_j x_j)^2} \\ \frac{-d_2 v_i^2}{(c_i+\sum_{j=1}^{n_i} d_j x_j)^2} \\ \vdots \\ \frac{-d_{n_i} v_i^2}{(c_i+\sum_{j=1}^{n_i} d_j x_j)^2} \end{bmatrix}$$

and

$$\nabla^2\zeta_i = \begin{bmatrix} \frac{2}{c_i+\sum_{j=1}^{n_i} d_j x_j} & -\frac{2v_i d_1}{(c_i+\sum_{j=1}^{n_i} d_j x_j)^2} & -\frac{2v_i d_2}{(c_i+\sum_{j=1}^{n_i} d_j x_j)^2} & \cdots & -\frac{2v_i d_{n_i}}{(c_i+\sum_{j=1}^{n_i} d_i x_j)^2} \\ -\frac{2v_i d_1}{(c_i+\sum_{j=1}^{n_i} d_j x_j)^2} & \frac{2v_i^2 d_1^2}{(c_i+\sum_{j=1}^{n_i} d_j x_j)^3} & \frac{2v_i^2 d_1 d_2}{(c_i+\sum_{j=1}^{n_i} d_j x_j)^3} & \cdots & \frac{2v_i d_1 d_{n_i}}{(c_i+\sum_{j=1}^{n_i} d_i x_j)^2} \\ -\frac{2v_i d_2}{(c_i+\sum_{j=1}^{n_i} d_j x_j)^2} & \frac{2v_i^2 d_1 d_2}{(c_i+\sum_{j=1}^{n_i} d_j x_j)^3} & \frac{2v_i^2 d_2^2}{(c_i+\sum_{j=1}^{n_i} d_j x_j)^3} & \cdots & \frac{2v_i d_2 d_{n_i}}{(c_i+\sum_{j=1}^{n_i} d_i x_j)^2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -\frac{2v_i d_{n_i}}{(c_i+\sum_{j=1}^{n_i} d_j x_j)^2} & \frac{2v_i^2 d_1 d_{n_i}}{(c_i+\sum_{j=1}^{n_i} d_j x_j)^3} & \frac{2v_i^2 d_2 d_{n_i}}{(c_i+\sum_{j=1}^{n_i} d_j x_j)^3} & \cdots & \frac{2v_i d_{n_i}^2}{(c_i+\sum_{j=1}^{n_i} d_i x_j)^2} \end{bmatrix}$$

Denoting

$$\gamma_i = \frac{1}{c_i + \sum_{j=1}^{n_i} d_j x_j},$$

then

$$\nabla^2 \zeta_i = 2\gamma_i \begin{bmatrix} -1 \\ \gamma_i d_1 v_i \\ \gamma_i d_2 v_i \\ \vdots \\ \gamma_i d_{n_i} v_i \end{bmatrix} [-1,\ \gamma_i d_1 v_i,\ \gamma_i d_2 v_i, \ldots, \gamma_i d_{n_i} v_i].$$

For $y^T \equiv [y_0, y_1, \ldots, y_{n_i}]$, we have,

$$y^T \nabla^2 \zeta_i(v_i, , x_1, \ldots, x_{n_i}) y = 2\gamma_i \left(-y_0 + \sum_{j=1}^{n_i} y_j \gamma_i d_j v_i \right)^2 \geq 0, \forall y \in \mathbb{R}^{n_i+1},$$

for each $(v_i, x_1, \ldots, x_{n_i}) \in \mathcal{Z}$. Therefore, ζ_i is convex on $\mathcal{Z}$. □

We are now able to prove that the lower bounding problem is a convex program.

Theorem 3 *The function*

$$\widetilde{g}^k(x, v) = \sum_{i=1}^{m} \frac{v_i^2}{r_i^k + \sum_{j \in I_i} (u_j^k + l_j^k) x_j}$$

in (29) *is convex over the constraint set of* $LB(P_k^3)$.

Proof This follows from Theorem 2, noting that (27) implies that $\gamma_i > 0, \forall i = 1, \ldots, m$ over the feasible region of $LB(P_k^3)$, and from the fact that the sum of convex functions is convex. □

As a consequence of this result, a global minimum for $LB(P_k^3)$ can be obtained as a stationary point of the objective function over its constraint set. This can be easily determined using a local nonlinear program solver, such as MINOS [31].

To continue the description of the branch–and–bound algorithm, we need to define the choice of the branching variable and the order in which the open nodes on the tree are investigated. As recommended in [36] and [35], the index s for the branching variable x_s can be chosen according to the following criterion, where x^{*k} solves Problem $LB(P_k^3)$:

$$s = \arg\max_{j=1,\ldots,n} \left\{ \left((u_j^k + l_j^k) x_j^{*k} - u_j^k l_j^k \right) - (x_j^{*k})^2 \right\}. \tag{34}$$

The branching procedure is based on the partitioning of the interval $[l_s^k, u_s^k]$ into two intervals, where x_s is the chosen variable for branching at the current node. This is done according to one of two rules. In the first rule, called **Branching Rule A**, the set $\Omega^k \equiv [l^k, u^k]$ is partitioned into $\Omega^{K+1} \equiv \left[l^{K+1}, u^{K+1}\right]$ and $\Omega^{K+2} \equiv \left[l^{K+2}, u^{K+2}\right]$, where

$$l^{K+1} = l^k, \qquad u_j^{K+1} = \begin{cases} u_j^k & \text{for } j \neq s \\ \frac{l_j^k + u_j^k}{2} & \text{for } j = s \end{cases} \tag{35}$$

$$l_j^{K+2} = \begin{cases} l_j^k & \text{for } j \neq s \\ \frac{l_j^k + u_j^k}{2} & \text{for } j = s \end{cases}, \qquad u^{K+2} = u^k. \tag{36}$$

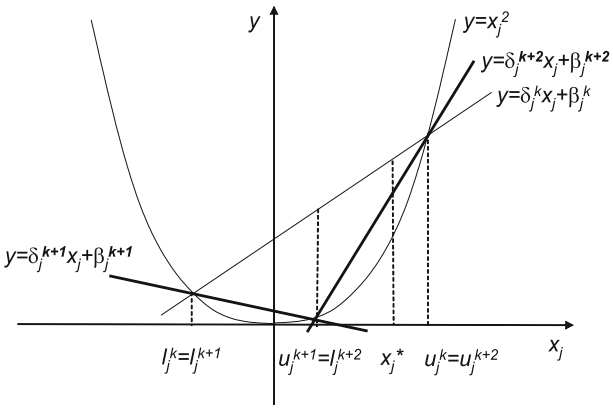

Fig. 2 Linear approximation of x_j^2 for the next pair of generated nodes

In an alternative **Branching Rule B**, we again select the branching index according to (34), but in lieu of bisecting the corresponding interval $[l_s^k, u_s^k]$, we cut it at the value x_s^{*k}. This is motivated by the fact that the error in the approximation of Fig. 1 is zero at the end points, and we would like to drive the error due to the linear approximation adopted in $LB(P_k^3)$ to zero. Hence, in this case, we partition Ω^k into Ω^{K+1} and Ω^{K+2}, where

$$\Omega^{K+1} = \left\{x : l_j^k \le x_j \le u_j^k,\ j = 1, \ldots, n,\ j \ne s,\ l_s^k \le x_s \le x_s^{*k}\right\}, \tag{37}$$

$$\Omega^{K+2} = \left\{x : l_j^k \le x_j \le u_j^k,\ j = 1, \ldots, n,\ j \ne s,\ x_s^{*k} \le x_s \le u_s^k\right\}. \tag{38}$$

By doing this, we expect to improve the affine approximation for x_s^2 in the nodes generated from the current node, as shown in Fig. 2, and consequently to reduce the gap between $g^k(x, v)$ and $f(x, v)$, leading to better lower bounds.

In either branching rule, note that whenever the argument $\{\cdot\}$ in (34) for $j = s$ is zero, then (x^{*k}, v^{*k}) is an optimal solution for P_k^3 and provides an upper bound $UB = f(x^{*k}, v^{*k})$ for the original problem P^3 and the node can be fathomed.

Let us consider now the task of finding upper bounds. An initial upper bound can be computed as a stationary point of P^3 given by (19–22). For each generated node, the upper bound can be updated as possible, by computing the value of the objective function in the unconstrained formulation P^2 given by (18) at x^{*k}, where (x^{*k}, v^{*k}) is the solution obtained for the lower bounding problem $LB(P_k^3)$.

Finally, the nodes may be investigated according to the least lower bound rule, that is, according to the criteria:

$$k \in \arg\min\left\{v(LB(P_t^3)) : t \in L\right\}, \tag{39}$$

where k is the label of the selected node for partitioning and L is the set of open nodes in the tree.

The steps of the branch–and–bound algorithm can now be formally summarized as follows. Let

k—be the index for the current lower bounding problem under analysis;
UB—the best known upper bound;
x_{inc}—the incumbent solution;

L—the queue of indices of subproblems created but not expanded;
K—the number of nodes generated apart from the root node;
$LB(P_k^3)$—the k^{th} lower bounding problem;
(x^{*k}, v^{*k})—the optimal solution obtained for $LB(P_k^3)$;
ϵ—the specified optimality tolerance ($\epsilon \geq 0$);
$v(.)$—the optimal value of Problem (.).

Branch–and–Bound algorithm (B&B)

0—(Initialization) Let $K = 0, k = 0, L = \{0\}$, and $\epsilon \geq 0$ (practically, we select $\epsilon = 10^{-6}$). Solve $LB(P_0^3)$. Find a stationary point for P_0^3 and let (x^*, v^*) be the solution obtained. Update $UB = f(x^*, v^*)$ and set $x_{inc} = x^*$.
1—(Choice of node) If $L = \emptyset$ then stop; otherwise find k by using (39). Set $L \leftarrow L - \{k\}$.
2—(Branching rule) Find a branching index s via (34).
3—(Solve,Update, and Queue) Set $i = 1$.

3.1 Define Ω^{K+i} according to Branching Rule **A** (35–36) or Branching Rule **B** (37–38). Solve Problem $LB(P_{K+i}^3)$.
3.2 If $v(LB(P_{K+i}^3)) < UB(1-\epsilon)$ then set $L \leftarrow L \bigcup \{K+i\}$ and go to Step 3.3; otherwise, go to Step 3.4.
3.3 Update UB accordingly to $UB \leftarrow \min\{f(x^{*K+i}, v^{*K+i}), UB\}$. If UB changes remove all indices $t \in L$ for which $v(LB(P_t^3)) \geq UB(1-\epsilon)$ and set $x_{inc} = x^{*K+i}$.
3.4 If $i = 2$ then set $K \leftarrow K + 2$ and go to Step 1; otherwise, let $i = 2$ and go to Step 3.1.

The next theorem establishes the convergence of the algorithm.

Theorem 4 *The algorithm* **B&B** (*run with* $\epsilon = 0$ *and with* Branching Rule **A** or **B**) *either terminates finitely with a global optimum to problem* P^3*, or else, an infinite branch-and-bound tree is generated, which is such that any accumulation point of the solution to the lower bounding problem along any infinite branch of the enumeration tree is a global optimum to Problem* P^3.

Proof The case of finite convergence is obvious from the validity of the lower and upper bounds derived by the algorithm. Hence, suppose that an infinite branch-and-bound tree is generated. Then, considering an infinite branch, there exists a branching variable index s that is selected infinitely often. Let $\mathcal{S}$ index the nodes along this infinite branch for which the branching index is given by s, and over some convergent subsequence indexed by $\mathcal{S}_1 \subseteq \mathcal{S}$, say, let

$$\{(x^{*k}, v^{*k}, l^k, u^k, g^k((x^{*k}, v^{*k}))\}_{\mathcal{S}_1} \rightarrow (x^*, v^*, l^*, u^*, \mathrm{V}^*). \tag{40}$$

We must show that (x^*, v^*) solves P^3. First (x^*, v^*) is clearly feasible to P^3. For Branching Rule **A**, by the bisection process (35–36), we have that

$$x_s^* = l_s^* = u_s^*. \tag{41}$$

Alternatively by the Branching Rule **B**, following the argument in [36], it is easily seen that

$$x_s^* = l_s^* \text{ or } x_s^* = u_s^*. \tag{42}$$

Moreover, by (27) and the branching index choice rule (34), we have,

$$\left[(u_s^k + l_s^k)x_s^{*k} - u_s^k l_s^k\right] - (x_s^{*k})^2 \geq \left[(u_j^k + l_j^k)x_j^{*k} - u_j^k l_j^k\right] - (x_j^{*k})^2 \geq 0, \quad \forall j = 1, \ldots, n, \quad \forall k \in \mathcal{S}_1. \tag{43}$$

Hence, taking limits in (43) as $k \to \infty$, $k \in \mathcal{S}_1$, and using (40) and either (41) or (42), we get

$$(x_j^*)^2 = (u_j^* + l_j^*)x_j^* - u_j^* l_j^*, \quad \forall j = 1, \ldots, n,$$

which implies that

$$\mathrm{V}^* = f(x^*, v^*) \geq v(P^3). \tag{44}$$

However, by the node selection rule (34), we have that $v[LB(P_k^3)] \leq v(P^3)$, $\forall k \in \mathcal{S}_1$, which in the limit as $k \to \infty$, $k \in \mathcal{S}_1$ yields via (40) that $\mathrm{V}^* \leq v(P^3)$. Hence, by (44), $\mathrm{V}^* = f(x^*, v^*) = v(P^3)$, and so (x^*, v^*) solves P^3. □

The RLT approach described in [8] can also be useful to generate lower-bounds for problem P_k^3. In this type of approach, new variables α_i, $i = 1, \ldots, m$, and y_j, $j = 1, \ldots, n$, are introduced such that

$$\alpha_i = \sum_{j \in I_i} y_j, \quad i = 1, \ldots, m. \tag{45}$$

$$x_j^2 = y_j, \quad j = 1, \ldots, n. \tag{46}$$

In order to get a convex program, the n constraints (46) are relaxed into $x_j^2 \leq y_j$. Furthermore, the following relaxed linear bound-factor constraints are introduced:

$$\left[(x_j - l_j^k)(u_j^k - x_j)\right]_L \geq 0, \quad \forall j = 1, \ldots, n,$$

where the operator $[\cdot]_L$ performs linearization of the product term $[\cdot]$ under the substitution $y_j = x_j^2$, $\forall j = 1, \ldots, n$. Therefore,

$$\left[(x_j - l_j^k)(u_j^k - x_j)\right]_L = \left[x_j u_j^k - l_j^k u_j^k + l_j^k x_j - x_j^2\right]_L \tag{47}$$

$$= (u_j^k + l_j^k)x_j - l_j^k u_j^k - y_j \tag{48}$$

and the following lower bounding problem $LB(P_k^3)$ is obtained:

$$(LB(P_k^4)): \ \min f_R(x, v, \alpha) = \sum_{i=1}^{m_0} \frac{v_i^2}{1 + \alpha_i} + \sum_{i=m_0+1}^{m} \frac{\left(\sum_{j \in I_i} a_{ij} x_j - b_i\right)^2}{1 + \alpha_i} \tag{49}$$

$$s.t.$$

$$v_i \geq \sum_{j=1}^{n} a_{ij} x_j - b_i, \quad i = 1, \ldots, m_0, \tag{50}$$

$$\sum_{j \in I_i} y_j = \alpha_i, \quad i = 1, \ldots, m, \tag{51}$$

$$x_j^2 \leq y_j, \quad j = 1, \ldots, n, \tag{52}$$

$$(u_j^k + l_j^k)x_j - l_j^k u_j^k \geq y_j, \quad j = 1, \ldots, n, \tag{53}$$

$$l^k \leq x \leq u^k \tag{54}$$

$$v_i \geq 0, \quad i = 1, \ldots, m_0. \tag{55}$$

Next, we show that this convex program is equivalent to problem $LB(P_k^3)$. To do this, we first eliminate the variables α_i by using (51). On the other hand, due to the definition of the

Table 1 Characteristics of test problems from Netlib

Name	m	n
Galenet	8	8
Itest2	9	4
Itest6	11	8
Bgprtr	20	34
Forest6	66	95
Woodinfe	35	89

lower-bound problem, the constraints (53) should hold as equalities in any optimal solution of the convex program. By replacing the variables y_j by their values from these equations, we get the problem $LB(P_k^3)$ with the further quadratic constraint

$$x_j^2 \leq (u_j^k + l_j^k)x_j - l_j^k u_j^k, \quad j = 1, \ldots, n.$$

Since these constraints are redundant by construction, they may be eliminated and the RLT lower bounding problem reduces to Problem $LB(P_k^3)$.

4 Computational experience

In order to test the performance of Algorithm **B&B** we report some computational results for a set of infeasible linear systems of the type:

$$\left\{ x \in \mathbb{R}^n : \sum_{j=1}^{n} a_{ij}x_j \leq b_i,\ i = 1, \ldots, m_0,\ \sum_{j=1}^{n} a_{ij}x_j = b_i,\ i = m_0+1, \ldots, m,\ l \leq x \leq u \right\},$$

where $A = [a_{ij}]$ is a real matrix of order $m \times n$ and $b = [b_i]$ is a real vector of size m. We consider two sets of problems. The first group is taken from a set of infeasible linear programming problems[2] selected from Netlib.[3] For the application of the algorithm in its current version, we have added lower and upper bounds, whenever not pre-specified. We used $l_i = 1$ and $u_i = 5$, for all i. As discussed later, other values for these lower and upper bounds have also been tested. Table 1 includes the number of constraints m and variables n of the chosen Netlib problems.

The second set consists of specially generated problems having some singular properties, described in [8], to which we have added a set of finite lower and upper bounds, given by $l_i = 1$ and $u_i = 5$ for all i. Table 2 summarizes the dimensions of these problems. Given that the proposed approach would typically be applied to an IIS from a certain parent problem, the sizes of problems considered in Tables 1 and 2 are adequate in practice.

All the tests have been performed on a Pentium IV (Intel) with Hyperthreading, CPU 3.0 GHz, 2Gb RAM, and operating system Linux. The method was implemented in the General Algebraic Modeling System (GAMS) [10] language (Rev 118 Linux/Intel) and the NLP solver MINOS (Version 5.5) has been used to compute the lower bounds required at each

[2] Test problems compiled by John W. Chinneck.

[3] The Netlib repository contains freely available software, documents, and databases. The repository is maintained by AT&T Bell Laboratories, the University of Tennessee, and the Oak Ridge National Laboratory, among other individual contributors.

Table 2 Dimension of test problems

Name	m	n
Pro6 to Pro10	20	10
Pro11 to Pro15	30	15
Pro16 to Pro20	40	20

Table 3 Computational results for Algorithm **B&B-A**

Problems	**B&B-A**							
	ρ	ND	CPU	ITER	INITUB	VALOPT	NDOPT	NUPDUB
Galenet	6	1	0.04	18	26.9608	26.9608	1	0
Itest2	6	20	0.17	110	0.9059	0.9059	1	0
Itest6	6	1	0.06	7	4,46,274,332.2501	4,46,274,332.2501	1	0
Bgprtr	1	1	0.01	60	67,358.9157	67,358.9157	1	0
Forest	1	32	0.25	598	65,213.6032	65,213.6032	1	0
Woodinfe	1	1	0.04	96	0.5060	0.5060	1	0
Prob4	6	6	0.05	220	157.6815	157.6815	1	0
Prob5	6	6	0.09	93	264.7228	262.8295	5	2
Prob6	6	7	0.09	316	315.1673	315.1673	1	0
Prob7	6	85	0.79	4,626	214.8726	214.8726	1	0
Prob8	6	1	0.03	54	2,187.0132	2,187.0132	1	0
Prob9	6	24	0.17	1,534	149.3484	149.3484	1	0
Prob10	6	34	0.30	1,379	250.8560	250.8560	1	0
Prob11	6	28	0.39	1,859	396.0405	396.0405	1	0
Prob12	6	3	0.03	199	1,130.5302	1,130.5302	1	0
Prob13	6	12	0.24	738	679.4526	679.4526	1	0
Prob14	6	6	0.21	426	756.9513	756.9513	1	0
Prob15	6	600	10.35	69,939	364.4502	364.4502	1	0
Prob16	6	23	0.66	2,775	1,121.5359	1,121.5359	1	0
Prob17	6	22	0.54	2,603	1,092.9802	1,092.9802	1	0
Prob18	6	8	0.29	917	1,738.2852	1,738.2852	1	0
Prob19	6	28	0.75	3,942	1,104.5614	1,104.5614	1	0
Prob20	6	105	3.10	15,972	944.7827	944.7827	1	0

node and for computing the upper bound at the root node. We present results for the algorithm described in Sect. 3 (**Branch–and–Bound Algorithm**). Since we implemented two Branching Rules **A** and **B**, we use the designations **B&B-A** and **B&B-B** to distinguish the respective results. The tolerance parameter ϵ was set to 10^{-6} in all problems. We limited the maximum number of nodes generated to 1,000 in the tree. Larger tolerances, $10^{-\rho}$, where $1 \leq \rho \leq 5$ and integer, have been used whenever the algorithm is unable to terminate in less than the maximum limit of 1,000 nodes for the tolerance 10^{-6}. These cases are displayed in the column of the parameter ρ, defined below. Each table reports the following information for each test problem:

ρ: maximum integer value belonging to {1, 2, 3, 4, 5, 6} for which the algorithm has been able to terminate in less than 1,000 nodes;

Table 4 Computational results for Algorithm **B&B-B**

Problems	**B&B-B**							
	ρ	ND	CPU	ITER	INITUB	VALOPT	NDOPT	NUPDUB
Galenet	6	1	0.03	17	26.9608	26.9608	1	0
Itest2	6	11	0.10	72	0.9059	0.9059	1	0
Itest6	6	1	0.04	7	4,46,274,332.2501	4,46,274,332.2501	1	0
Bgprtr	4	966	5.30	3,985	67,358.9157	67,358.9157	1	0
Forest	1	95	0.88	1,071	65,213.6032	65,213.6032	1	0
Woodinfe	1	1	0.02	96	0.5060	0.5060	1	0
Prob4	6	5	0.05	189	157.6815	157.6815	1	0
Prob5	6	5	0.07	78	264.7228	262.8295	4	1
Prob6	6	5	0.10	208	315.1673	315.1673	1	0
Prob7	6	68	0.60	3,572	214.8726	214.8726	1	0
Prob8	6	1	0.07	46	2,187.0132	2,187.0132	1	0
Prob9	6	20	0.26	1,192	149.3484	149.3484	1	0
Prob10	6	29	0.28	1,244	250.8560	250.8560	1	0
Prob11	6	26	0.33	1,596	396.0405	396.0405	1	0
Prob12	6	3	0.06	190	1,130.5302	1,130.5302	1	0
Prob13	6	12	0.16	811	679.4526	679.4526	1	0
Prob14	6	6	0.07	420	756.9513	756.9513	1	0
Prob15	6	642	5.67	24,340	364.4502	364.4502	1	0
Prob16	6	26	0.59	2,916	1,121.5359	1,121.5359	1	0
Prob17	6	27	0.75	2,914	1,092.9802	1,092.9802	1	0
Prob18	6	8	0.23	790	1,738.2852	1,738.2852	1	0
Prob19	6	25	0.67	3,455	1,104.5614	1,104.5614	1	0
Prob20	6	199	5.28	28,638	944.7827	944.7827	1	0

ND: total number of nodes in the tree;
CPU: total CPU time in seconds;
ITER: total number of MINOS iterations;
INITUB: initial upper bound (obtained at the root node);
VALOPT: optimal value;
NDOPT: node at which the optimal solution was obtained;
NUPDUB: number of upper bound updates.

The results show that the local nonlinear programming algorithm employed at the root node always computes a strong upper bound, which is often the global minimum of the fractional program, although this yet requires some enumeration (see the column for ND) to verify optimality. When the optimal solution is not obtained at the root node, then it is found almost towards the end of the tree search, and the number of upper bound updates is quite small. Comparing the performance of the two branching rules, there is no great evidence of supremacy of one with respect to the other. As far as the computational time, number of iterations and number of nodes are concerned, the best performance for each problem alternates between both algorithms. However, for approximately 74% of the problems, a fewer number of iterations (ITER) resulted with **Branching Rule A** than with **Branching Rule B**. In addition, Problem Bgprtr was solved with a tolerance $\epsilon = 10^{-4}$ using **Branching Rule**

Table 5 Computational results for Algorithm **B&B-A**

Problems	**B&B-A**								
	t	ρ	ND	CPU	ITER	INITUB	VALOPT	NDOPT	NUPDUB
Itest2	1	1	1	0.02	21	9.0000	9.0000	1	0
	1	2	2	0.03	26	9.0000	9.0000	1	0
	1	3	3	0.04	30	9.0000	9.0000	1	0
	1	4	5	0.10	38	9.0000	9.0000	1	0
	1	5	7	0.17	46	9.0000	9.0000	1	0
	1	6	8	0.06	50	9.0000	9.0000	1	0
Itest2	5	1	1	0.04	35	0.8999	0.8999	1	0
	5	2	9	0.11	114	0.8999	0.8999	1	0
	5	3	15	0.14	151	0.8999	0.8999	1	0
	5	4	23	0.23	198	0.8999	0.8999	1	0
	5	5	33	0.25	243	0.8999	0.8999	1	0
	5	6	42	0.31	277	0.8999	0.8999	1	0
Itest2	50	1	19	0.09	718	0.7418	0.7418	1	0
	50	2	37	0.28	1,242	0.7418	0.7418	1	0
	50	3	60	0.29	1,922	0.7418	0.7418	1	0
	50	4	83	0.40	2,499	0.7418	0.7418	1	0
	50	5	111	0.59	3,150	0.7418	0.7418	1	0
	50	6	135	0.62	3,657	0.7418	0.7418	1	0
Itest2	500	1	34	0.27	1,900	0.7418	0.7418	1	0
	500	2	64	0.56	2,968	0.7418	0.7418	1	0
	500	3	96	0.72	4,027	0.7418	0.7418	1	0
	500	4	117	0.61	4,657	0.7418	0.7418	1	0
	500	5	144	1.13	5,376	0.7418	0.7418	1	0
	500	6	151	1.08	5,529	0.7418	0.7418	1	0
Bgprtr	1	1	1	0.08	42	13,099.7326	13,099.7326	1	0
	1	2	16	0.19	189	13,099.7326	13,099.7326	1	0
	1	3	56	0.33	534	13,099.7326	13,099.7326	1	0
	1	4	121	0.69	982	13,099.7326	13,099.7326	1	0
	1	5	271	1.31	1,986	13,099.7326	13,099.7326	1	0
	1	6	624	3.61	4,209	13,099.7326	13,099.7326	1	0
Bgprtr	5	1	10	0.13	407	656.4412	656.4412	1	0
	5	2	–	–	–	–	–	–	–
Bgprtr	50	1	14	0.15	874	3.6797	3.6797	1	0
	50	2	–	–	–	–	–	–	–
Bgprtr	500	1	1	0.05	298	0.0070	0.0070	1	0
	500	2	1	0.05	298	0.0070	0.0070	1	0
	500	3	–	–	–	–	–	–	–
Woodinfe	1	1	1	0.03	119	26.5921	26.5921	1	0
	1	2	1	0.03	119	26.5921	26.5921	1	0

Table 5 continued

Problems	**B&B-A**								
	t	ρ	ND	CPU	ITER	INITUB	VALOPT	NDOPT	NUPDUB
	1	3	107	0.81	1,807	26.5921	26.5921	1	0
	1	4	352	2.42	4,885	26.5921	26.5921	1	0
	1	5	554	3.44	7,004	26.5921	26.5921	1	0
	1	6	628	3.67	7,637	26.5921	26.5921	1	0

B, while for the **Branching Rule A**, the maximum number (1,000) of nodes allowed was reached without satisfying the stopping criterion. Actually, for this latter rule, the tolerance had to be increased to 10^{-1} for the algorithm to be able to terminate. Another relatively difficult problem in this test-bed is Prob15, for which **B&B-B** again yielded a significantly superior performance in terms of computational time and iterations, despite the number of nodes being larger. Given that the remaining problems (except for Prob20) took less than 1 cpu second to be solved, we recommend **B&B-B** on an overall basis. Also, the root node algorithm evidently offers a strong heuristic for this class of problems.

In a second experiment, we performed a sensitivity analysis for the values of the upper bounds that we have added to the last five NETLIB problems (Galenet has its own lower and upper bounds and is not included in these tests). This was done by choosing the lower and upper bounds for each variable x_i such that $l_i = 0$ and $u_i = t$, for $t = 1, 5, 50, 500$. Furthermore, we ran the algorithm for different tolerances $10^{-\rho}$, where $\rho \in \{1, 2, 3, 4, 5, 6\}$, and where this tolerance was used to end the tree search. The results of this experiment are displayed in Tables 5 and 6, where "-" means that the algorithm was unable to terminate for a tolerance smaller than or equal to $10^{-\rho}$, for a ρ given in this row. We have not included in these tables the performance of the algorithms for Problem Itest6, as termination always occurred at the initial node itself with 7 iterations for a tolerance of 10^{-6}, independent of the chosen lower and upper bounds. Likewise, we do not display results for Problem Woodinfe for $t \geq 5$, or for Problem Forest for $t \geq 1$, because the procedures failed to terminate, although we did obtain a successful termination for $l_i = 1$ and $u_i = 5, \forall i$, with $\rho = 1$ (see Tables 3 and 4).

For the remaining NETLIB problems the algorithm appears to be quite efficient whenever the amplitude of the bounds is equal to one. As expected, the computational effort increases with a decrease of the value of the tolerance. The performance of the algorithm for amplitudes 50 and 500 is usually worse than for the first case, but sometimes, as with Problem Bgprtr, wider bounds yield an improved solution (at the initial node itself), which thereby induces a faster termination. The algorithm successfully terminated for a tolerance 10^{-1} or 10^{-2} but was unable to guarantee a global minimum for smaller values of the tolerance. As expected, the set Ω has a significant impact on the efficiency of the branch-and-bound algorithm. Different strategies for defining tighter bounds for the set Ω as implied by the model structure and underlying application may lead to a better performance for the algorithm and should be studied in future research.

Finally we have compared the optimal value of the problem with the value of the objective function obtained using a local solver. We used the function *fmincon* from Matlab applied to the nonconvex formulation (18) with box constraints. We adopted the same bounds as those used in the branch–and–bound approach. In Table 7 we present in column $VALOPT$ the

Table 6 Computational results for Algorithm **B&B-B**

Problems	**B&B-B**								
	t	ρ	ND	CPU	ITER	INITUB	VALOPT	NDOPT	NUPDUB
Itest2	1	1	1	0.04	15	9.0000	9.0000	1	0
	1	2	1	0.04	15	9.0000	9.0000	1	0
	1	3	2	0.05	18	9.0000	9.0000	1	0
	1	4	3	0.06	20	9.0000	9.0000	1	0
	1	5	4	0.07	22	9.0000	9.0000	1	0
	1	6	4	0.06	22	9.0000	9.0000	1	0
Itest2	5	1	1	0.03	30	0.8999	0.8999	1	0
	5	2	10	0.12	112	0.8999	0.8999	1	0
	5	3	16	0.12	158	0.8999	0.8999	1	0
	5	4	28	0.14	228	0.8999	0.8999	1	0
	5	5	33	0.19	258	0.8999	0.8999	1	0
	5	6	40	0.20	304	0.8999	0.8999	1	0
Itest2	50	1	24	0.11	471	0.7418	0.7418	1	0
	50	2	109	0.58	1, 824	0.7418	0.7418	1	0
	50	3	150	0.61	2, 338	0.7418	0.7418	1	0
	50	4	164	0.91	2, 522	0.7418	0.7418	1	0
	50	5	180	0.96	2, 680	0.7418	0.7418	1	0
	50	6	197	0.96	2, 837	0.7418	0.7418	1	0
Itest2	500	1	51	0.44	1, 289	0.7418	0.7418	1	0
	500	2	212	0.96	4, 627	0.7418	0.7418	1	0
	500	3	243	1.18	5, 163	0.7418	0.7418	1	0
	500	4	262	1.32	5, 386	0.7418	0.7418	1	0
	500	5	292	1.57	5, 676	0.7418	0.7418	1	0
	500	6	323	1.35	5, 957	0.7418	0.7418	1	0
Bgprtr	1	1	1	0.03	43	13, 099.7326	13, 099.7326	1	0
	1	2	7	0.08	70	13, 099.7326	13, 099.7326	1	0
	1	3	7	0.08	70	13, 099.7326	13, 099.7326	1	0
	1	4	15	0.15	111	13099.7326	13, 099.7326	1	0
	1	5	15	0.14	111	13, 099.7326	13, 099.7326	1	0
	1	6	17	0.17	122	13, 099.7326	13, 099.7326	1	0
Bgprtr	5	1	3	0.03	186	656.4412	656.4412	1	0
	5	2	127	0.61	3, 544	656.4412	656.4412	1	0
	5	3	–	–	–	–	–	–	–
Bgprtr	50	1	63	0.65	2, 460	3.6797	3.6797	1	0
	50	2	251	1.68	9, 933	3.6797	3.6797	1	0
	50	3	–	–	–	–	–	–	–
Bgprtr	500	1	1	0.03	269	0.0070	0.0070	1	0
	500	2	1	0.05	269	0.0070	0.0070	1	0
	500	3	63	0.55	1, 461	0.0070	0.0070	1	0
	500	4	–	–	–	0.0070	0.0070	1	0

Table 6 continued

Problems	B&B-B								
	t	ρ	ND	CPU	ITER	INITUB	VALOPT	NDOPT	NUPDUB
Woodinfe	1	1	1	0.03	120	26.5921	26.5921	1	0
	1	2	1	0.03	120	26.5921	26.5921	1	0
	1	3	111	0.54	1780	26.5921	26.5921	1	0
	1	4	250	1.65	3300	26.5921	26.5921	1	0
	1	5	452	2.71	5635	26.5921	26.5921	1	0
	1	6	556	3.70	6, 660	26.5921	26.5921	1	0

Table 7 Global versus local solvers

Problems	VALOPT	NLP-con
Galenet	26.9608	26.9608
Itest2	0.9059	0.9059
Itest6	4,46,274,332.2501	4,46,274,332.2501
Bgprtr	67,358.9157	67,358.9157
Forest	65,213.6032	65,213.6032
Woodinfe	0.5060	**1.0000**
Prob4	157.6815	157.6815
Prob5	264.7228	**272.6953**
Prob6	315.1673	**354.4125**
Prob7	214.8726	214.8726
Prob8	2,187.0132	**2,387.1776**
Prob9	149.3484	149.3484
Prob10	250.8560	250.8560
Prob11	396.0405	396.0405
Prob12	1,130.5302	**1,152.3014**
Prob13	679.4526	679.4526
Prob14	756.9513	756.9513
Prob15	364.4502	**392.3507**
Prob16	1,121.5359	**1,157.0987**
Prob17	1,092.9802	**1,096.6078**
Prob18	1,738.2852	**1,825.2504**
Prob19	1,104.5614	**1,125.3224**
Prob20	944.7827	944.7827

optimal value obtained by the branch–and–bound procedure and in column $NLP—con$ the value of the solution obtained using $fmincon$ from Matlab.

The results show that in 10 of the 23 problems the value of the solution obtained using the local solver was not the global optimal solution (these are displayed in bold in Table 7). In our opinion, this experience supports our decision to design a branch-and-bound global optimization algorithm to find the minimum Frobenius norm correction.

5 Conclusions

This paper introduces a fractional programming formulation for a zero preserving correction of general inconsistent systems of linear constraints. Such systems analyzed might typically be IIS substructures inherent within some parent model. A branch-and-bound (**B&B**) algorithm is discussed for finding a global minimum to this optimization problem. This procedure is based on the linearization of the functions in each denominator of the fractional terms in the objective function. The tree search procedure developed is based on partitioning the domain of the original variables, where two different branching strategies are proposed, and also incorporates techniques for obtaining upper bounds at the root node as well as at each node of the enumeration tree. Theoretical convergence of the proposed **B&B** algorithm has been established. Computational experience reported on problems with (m, n) ranging up to (96, 95) shows that the **B&B** algorithm is effective for processing the equivalent reformulated fractional programming problem (particularly with **Branching Rule B**), and also, the root node upper bounding procedure offers a strong heuristic that can be efficiently applied for analyzing relatively larger problems than those considered herein.

Acknowledgements Research partially supported by project *FCT-POCI*/MAT/56704/2004, Portugal, *FCT-POCI*/MAT/56704/2004, Portugal and supported by the *National Science Foundation*, under Grant Number CMMI - 0552676.

References

1. Amaldi, E., Kann, V.: The complexity and approximability of finding maximum feasible subsystems of linear relations. Theor. Comput. Sci. **147**, 181–210 (1995)
2. Amaldi, E., Pfetsch, M.E., Trotter, J.L.E.: On the maximum feasible subsystems, IISs and IIS-hypergraphs. Math. Ser. A **95**, 533–554 (2003)
3. Amaral, P.: Contribuições para o estudo de sistemas lineares inconsistentes. Ph.D Disertation, Faculty of Science and Technology, UNL, Lisbon, Portugal (2001) (in Portuguese)
4. Amaral, P., Barahona, P.: About infeasibility in the constraints of a linear model. Ricerca Operativa **92**, 49–67 (1999)
5. Amaral, P., Barahona, P.: On optimal correction of inconsistent linear constraints. In: Hentenryck, P.V. (ed). Principles and Practice of Constraint Programming, CP'2002, NY, (Procs.), Lecture Notes in Computer Science, vol. 2470, pp. 33–46, Springer, Berlin (2002)
6. Amaral, P., Barahona, P.: Connections between the total least squares and the correction of an infeasible system of linear inequalities. Linear Algebra Appl. **395**, 191–210 (2005)
7. Amaral, P., Barahona, P.: A framework for optimal correction of inconsistent linear constraints. Constraints **10**, 67–86 (2005)
8. Amaral, P., Júdice, J., Sherali, H.D.: A reformulation–linearization–convexification algorithm for optimal correction of an inconsistent system of linear constraints. Comput. Oper. Res. **35**, 1494–1509 (2008)
9. Bazaraa, M.S., Sherali, H.D., Shetty, C.M.: Nonlinear Programming: Theory and Algorithms. 3rd edn. Wiley, New York, NY (2006)
10. Brooke, A., Kendrick, A., Meeraus, A., Raman, R.: GAMS—A User's Guide. http://www.gams.com/docs/document.htm
11. Chakravarti, N.: Some results concerning post-infeasibility analysis. Eur. J. Oper. Res. **73**, 139–143 (1994)
12. Chinneck, J.W.: Minos(ii):Infeasibility analysis using MINOS. Comput. Oper. Res. **21**, 1–9 (1994)
13. Chinneck, J.W.: An effective polynomial-time heuristic for the minimum-cardinality IIS set-covering problem. Ann. Math. Artif. Intell. **17**, 127–144 (1996)
14. Chinneck, J.W.: Finding a useful subset of constraints for analysis in an infeasible linear program. INFORMS J. Comput. **9**, 164–174 (1997)
15. Chinneck, J.W., Dravnieks, E.W.: Locating minimal infeasible constraint sets in linear programs. ORSA J. Comput. **3**, 157–168 (1991)
16. Chinneck, J.W., Saunders, M.A.: Minos(iis) version 4.2: Analyzing infeasibilities in linear programs. Eur. J. Oper. Res **81**, 217–218 (1995)

17. CPLEX: http://www.ilog.com/products/cplex/
18. Eremin, I.I.: Duality for nonproper problems of linear and convex programming, Dok1. Akad Nauk SSSR **256**(252), 272–276 (1981)
19. Falk, J.E., Soland, R.M.: Algorithm for separable nonconvex programming problems. Manag. Sci. Ser. A Theory **15**, 550–569 (1969)
20. Greenberg, H.J.: Computer-assisted analysis for diagnosing infeasible or unbounded linear programs. Math. Program. Study **31**, 79–97 (1987)
21. Greenberg, H.J.: Consistency, redundancy and implied equalities in linear systems. Ann. Math. Artif. Intell. **17**, 37–83 (1993)
22. Greenberg, H.J.: Enhancements of analyse: A computer-assisted analysis system for mathematical programming models and solutions. ACM Trans. Math. Softw. **19**, 233–256 (1993)
23. Greenberg, H.J.: How to analyze the results of linear programs- Part 3: Infeasibility Diagnoses. Interfaces **23**, 120–139 (1993)
24. Greenberg, H.J., Murphy, F.H.: Approaches to diagnosing infeasible linear programs. ORSA J. Comput. Study **31**, 79–97 (1991)
25. Guieu, O., Chinneck, J.W.: Analyzing infeasible mixed-integer and integer linear programs. INFORMS J. Comput. **11**, 63–77 (1999)
26. Holzbaur, C., Menezes, F., Barahona, P.: Defeasibility in CLP(Q) through generalised slack variables. In: Freuder, E.C. (ed.). Proceedings of CP´96, 2nd Int. Conf. in Principles and Practice of Constraint Programming. Lecture Notes in Computer Science, vol. 1118, pp. 209–223. Springer–Verlag, Berlin, (1996)
27. Horst, R., Tuy, H.: Global Optimization: Deterministic Approaches. 3rd edn. Springer-Verlag, Berlin (1996)
28. Horst, R., Pardalos, P., Thoai, N.: Introduction to global optimization. 2nd ed. Nonconvex Optimization and Its Applications, 48. Kluwer Academic Publishers, Dordrecht (2000)
29. LINDO: http://www.lindo.com/
30. Loon, J.N.M.: Irreducible inconsistent systems of linear inequalities. Eur. J. Oper. Res. **8**, 282–288 (1981)
31. Murtagh, B.A., Saunders M.A., Murray, W., Gill, P.E., Raman, R., Kalvelagen, E.: MINOS- NLP solver from Stanford University. http://www.gams.com/docs/document.htm
32. Pena, J.: Understanding the geometry of infeasible perturbations of a conic linear system. SIAM J. Optim. **10**, 534–550 (2000)
33. Renegar, J.: Some perturbation theory for linear programming. Math. Program. **65**, 73–91 (1994)
34. Roodman, G.M.: Post-infeasibility analysis in linear programming. Manag. Sci. **25**, 916–922 (1979)
35. Sherali, H.D., Adams, W.P.: A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems. Kluwer Academic Publishers, Dordrecht (1999)
36. Sherali, H.D., Tuncbilek, C.H.: A global optimization algorithm for polynomial programming problems using a reformulation-linearization technique. J. Glob. Optim. **2**, 101–112 (1992)
37. Tamiz, M., Mardle, S.J., Jones, D.F.: Detecting IIS in infeasible linear programmes using techniques from goal programming. Comput. Oper. Res. **23**, 113–119 (1996)
38. Vatolin, A.A.: Parametric approximation of inconsistent systems of linear equations and inequalities. Seminarber., Humboldt-Univ. Berlin Sekt. Math. **81**, 145–154 (1986)
39. Vatolin A.A.: Solvability sets and correction of saddle functions and inequality systems. Ural Branch Acad. Sci. USSR, Sverdlovsk (1989) (in Russian)
40. Vatolin, A.A.: An LP-based algorithm for the correction of inconsistent linear equation and inequality systems. Optimization **24**, 157–164 (1992)
41. Vera, J.R.: Ill-posedness and the complexity of deciding existence of solutions to linear programs. SIAM J. Optim. **6**, 549–569 (1996)
42. Wang, H.F., Huang, C.S.: Inconsistent structures of linear systems. Int. J. Gen. Syst. **21**, 65–81 (1992)