Journal of Global Optimization **9:** 153–167, 1996.

Computational Experience Using an Edge Search Algorithm for Linear Reverse Convex Programs

STEPHEN E. JACOBSEN
Electrical Engineering Department, University of California, Los Angeles, California 90024, U.S.A. (email: jacobsen@ee.ucla.edu)

KHOSROW MOSHIRVAZIRI
Information Systems Department, California State University, Long Beach, California, 90840 and Electrical Engineering Department, University of California, Los Angeles, CA 90024, U.S.A. (email: moshir@csulb.edu)

(Received: 8 November 1994; accepted: 23 January 1996)

Abstract. This paper presents computational experience with a rather straight forward implementation of an edge search algorithm for obtaining the globally optimal solution for linear programs with an additional reverse convex constraint. The paper's purpose is to provide a collection of problems, with known optimal solutions, and performance information for an edge search implementation so that researchers may have some benchmarks with which to compare new methods for reverse convex programs or concave minimization problems. There appears to be nothing in the literature that provides computational experience with a basic edge search procedure. The edge search implementation uses a depth first strategy. As such, this paper's implementation of the edge search algorithm is a modification of Hillestad's algorithm [11]. A variety of test problems is generated by using a modification of the method of Sung and Rosen [20], as well as a new method that is presented in this paper. Test problems presented may be obtained at ftp://newton.ee.ucla.edu/nonconvex/pub/.

Key words: Reverse convex programs, nonconvex optimization, global optimization, test problem generation, linear programming, nonlinear programming, computational experiments.

1. Introduction

Many problems of engineering design give rise to nonconvex global optimization problems. In this paper, we restrict our attention to the linear reverse convex programming problem, denoted by LRCP. This problem is defined to be

$$\min \{ c^\top x \mid x \in P,\ g(x) \leq 0 \}$$

where P is a convex polytope in R^n and $g : R^n \longrightarrow R^1$ is a concave function. The constraint is called reverse convex since the direction of the inequality is the reverse of what is needed to have a convex programming problem. The principle difficulty with LRCP is the fact that the reverse convex constraint generally produces a nonconvex feasible region with non-global local minima and, often, a disconnected feasible region. The computational complexity of this class of problems can be seen by noting that the 0–1 linear integer programming problem can be rewritten

as $\min\{c^\top x \mid x \in P,\ x \in I_n,\ x^\top(e-x) \leq 0\}$, a member of the class LRCP, where e denotes a vector of all ones and I_n is a unit hypercube in R^n. Similarly, the concave minimization problem $\min\{f(x) \mid x \in P\}$ can be rewritten as $\min\{\eta \mid -\eta + f(x) \leq 0,\ LB \leq \eta \leq UB,\ x \in P\}$, a member of the class LRCP. Of course, the lower bound, LB, must be at least as small as the optimal function value of the original concave minimization problem. A specific form of LRCP was studied by Bansal and Jacobsen [3, 4] in the context of network flow capacity optimization under economies-of-scale. Hillestad and Jacobsen [10, 9] showed that the convex hull of the feasible region is a convex polytope and, as a result, a global solution lies on an edge of P. In [18] Rosen developed an iterative procedure, based upon the linearization of g, that converges to a local solution. Avriel and Williams [1, 2] developed a similar procedure in the context of engineering design. Ueing [24] developed a branch and bound procedure, for the case of multiple reverse convex constraints, by solving a number of convex programming problems. In [21], an algorithm is developed that is based upon an alternating sequence of linear programs and concave minimization problems. Hillestad [11] developed an edge search algorithm for finding an optimal solution for (LRCP). The algorithm works within $P \sim G$ to find, in a systematic fashion, intersections of $\{x \mid g(x) = 0\}$ with the edges of P. Subsequently, Hillestad and Jacobsen [9] developed an algorithm that works within the feasible region, $P \cap G$. This latter approach was subsequently generalized by Tuy [22]. Of course, there have been several algorithms developed for LRCP and we refer the reader to the text of Horst and Tuy [12] for a fairly complete bibliography of such methods.

We rewrite problem (LRCP) as follows:

$$\begin{aligned} &\textit{Minimize}\ \ c^\top x \\ &\textit{Subject to:}\quad Ax \leq b \\ &\qquad\qquad\ g(x) \leq 0 \qquad\qquad \text{(LRCP)} \qquad\qquad (1) \\ &\qquad\qquad\ \ x \geq 0 \end{aligned}$$

where $g{:}R^n \to R^1$ is concave, A is $m \times n$ and, of course, c and b are vectors of order n and m, respectively. Let $P = \{x | Ax \leq b, x \geq 0\}$, $G = \{x | g(x) \leq 0\}$ and denote the feasible region by $F = P \cap G$. We assume that $F \neq \phi$ and that P is bounded. Furthermore, we assume that there is a unique optimal solution for the associated linear program, $x^\circ = \textit{Argmin}\ \{c^\top x |\ x \in P\}$, and that $x^\circ \notin G$. It is well known that an optimal solution for (LRCP) lies on an edge of P and on the boundary of G (e.g., see [10, 9]). Thus, it is sufficient to find all the intersections of the surface $\{x \mid g(x) = 0\}$ with the edges of P and then to choose the best among them as an optimal solution.

However, with respect to the above methods, there is nothing in the literature that demonstrates numerical efficiency of such methods, in comparison with the basic notion of edge search for solving LRCP. It is to the latter point that this paper is addressed. We describe below our implementation of an edge search algo-

rithm, our development of LRCP test problems, and our solution information for each of these test problems. Our point is not the development of a new algorithm; rather, our purpose is to set forth the results of a relatively efficient implementation of edge search and to make the numerical results known to the research community. Additionally, these problems are available to the research community at `ftp://newton.ee.ucla.edu/nonconvex/pub/`; as a result, researchers have at their disposal actual LRCP problems that have been solved by an edge search implementation and will be able to demonstrate the relative numerical efficiency of newly proposed algorithms.

2. Description of the Algorithm

Since the purpose of this paper is to present LRCP problems that have been solved by an edge search implementation, it is important to fully describe that implementation. Hillestad [11] describes a *breadth first* strategy in his edge search procedure. That is, from x°, the solution of the associated linear program, each of the neighbors, with higher objective values, is visited; then, from each of these neighbors, visits are made to their corresponding neighbors; the method continues in this fashion. Vertices are fathomed along the way and the method produces an optimal solution on an edge of P. Our implementation is a *depth first* strategy; that is, a sequence of neighboring vertices, starting with x°, is followed. Each vertex in this sequence has a larger value of the objective function; vertices are added to this sequence until a vertex is fathomed. At this point, we return to the predecessor of the fathomed vertex and similarly generate a new sequence of such vertices, starting from the predecessor vertex, that does not visit previously visited vertices. The intuitive justification for a depth first strategy is that it seems likely to produce a good upper bound more quickly than a breadth first strategy may. Furthermore, the process of fathoming vertices, as well as the selection of vertices to be added to a specific sequence (or search path), is done according to certain heuristic rules and criteria to be described shortly. As in Hillestad's algorithm, upper bounds are updated at various points and are used to fathom vertices.

Let V denote the set of vertices of P; for each $x \in V$ let $NV(x)$ denote the set of neighboring vertices. For each vertex $x \in V$, let $N(x) = \{\, y \in NV(x) \mid c^\top (y - x) > 0 \,\}$. That is, $N(x)$ is the set of neighbors with larger objective values. Also, for each vertex x of P, we let $u, v \in V$ denote neighboring vertices of x with the properties $x \in N(u)$ and $v \in N(x)$. In the algorithm's description it is to be understood that whenever x is updated, the predecessor of x, is also updated. Let x° denote the unique optimal solution for the linear program and we assume, of course, that x° is not feasible for the reverse convex constraint. Set the initial upper bound to be UB. In the statement of the algorithm that follows, we assume that we have at hand an initial estimate of an optimal solution and a finite upper bound. Of course, we may take $\hat{x} = Argmax\,\{c^\top x \mid Ax \leq b,\ x \geq 0\,\}$, and set $UB = c^\top \hat{x}$. In what follows, $\mathcal{F}$ denotes the set of fathomed vertices. Initially, $\mathcal{F} = \phi$. At various

TABLE I. Data structure of the book-keeping process

Structure of the Recorded Data				
Vertex	x°	.....	x^i	.....
$g(x)$	$g(x^\circ)$		$g(x^i)$	
r	–		r_i	
s	–	.....	s_i	.....
ℓ	$\ell_\circ$		ℓ_i	
L	$L_\circ$		L_i	

points in the procedure, we will have a pair of vertices, u and x, with the property that $g(u) > 0$ and $g(x) \leq 0$. When this occurs, we need to find the intersection of the surface $\{ y \mid g(y) = 0 \}$ with the edge $[u, x]$. Note that this may easily be accomplished by solving the one-dimensional problem

$$\min \{ \alpha \in [0, 1] \mid g(u + \alpha(x - u) = 0. \}$$

Note that an optimal α, say $\alpha^\star$, is unique. We call the point $\bar{x} = u + \alpha^\star(x - u)$ a *first point.* A sufficient condition for $\bar{x}$ to be a strict local minimum is that $\bar{x}$ uniquely solves the linear program that arises when g is linearized about $\bar{x}$ [10]. However, not all first points are local minima.

2.1. Description of the Implementation

When at a vertex x, such that $N(x) \sim \mathcal{F} \neq \phi$, we introduce into the basis the $v \in N(x)$ with the smallest associated updated cost coefficient. This heuristic choice is based upon the intuitive notion that we would like to increase the objective function as little as possible in order to get a good upper bound. When at a vertex x we denote the number of elements of $N(x)$ by L, $|N(x)| = L$, and we denote by ℓ the number of vertices of $N(x)$ that have already been visited from x (initially, of course, $\ell = 0$). Each time a new element of $N(x)$ is chosen, the counter ℓ is augmented by one, $\ell = \ell + 1$. At such a pivot, we keep track of the index of the column of the simplex tableau that is introduced into the basis and the index of the column that is removed from the basis; we denote these two numbers by s and r, respectively. This is the minimal and most essential set of parameters whose values must be stored for each visited vertex that has not been eliminated from further consideration. Efficient data handling and storage management of data are instrumental for the success of any edge search implementation. Table I is a schematic of the book-keeping process.

To proceed with the description of our procedure, we will first need the following definitions:

DEFINITION (1). A vertex x is said to be *exhausted* or *fathomed by fathoming criterion type one* if there is no unsearched edge emanating from x that leads to a better vertex of P with respect to upward movement of the objective function, $N(x) = \phi$.

DEFINITION (2). A vertex x is said to be fathomed by *fathoming criterion type two* if function g evaluated at this point is positive and if the objective value at x, denoted by z, is greater than $z^\star$, where $z^\star = c^\top x^\star$ and $x^\star$ is the current best solution to (LRCP). That is, $x \in G$ and $c^\top x > UB = c^\top x^\star$.

2.2. Further Elimination of Vertices: Fathoming Criterion 3

Let $x^\star$ denote the current best local solution to (LRCP) and $H_\star$ denote the bounding plane at $x^\star$, namely a translation of $H_\circ$ to the point $x^\star$; $H_\star = \{ x \mid c^\top x = c^\top x^\star \}$. Moreover, let x be the current vertex under consideration such that $g(x) > 0$ and $c^\top x < c^\top x^\star$. Ordinarily, x is subject to further search. However, we first perform the following test on x and then decide what action to take next. If x fails the test, which we call *fathoming criterion type three*, then it will not be fathomed. Let the points y^k, $k = 1, \ldots, L$, denote the intersection of the extended rays emanating from x to all vertices in $N(x)$ with the hyperplane $H_\star$. That is

$$y^k = x + \alpha_k (u^k - x), \quad u^k \in N(x), \quad \alpha_k > 0.$$

DEFINITION (3). Vertex x is said to be fathomed by *fathoming criterion type three* if $g(x) > 0$ and $g(y^k) > 0$ for all $k = 1, \ldots, L$.

Let z^k, $k = 1, \ldots, p$ denote the intersection points of distinct upward (with respect to the objective function) paths, initiated from x, with $H_\star$, for some integer p. Clearly,

$$\{ z^k, \ k = 1, \ldots, p \} \subset V(H_\star \cap P) \, .$$

Let y^k be defined as above and let $\mathrm{C} = conv \{ x^\star; \ y^k, \ k = 1, \ldots, L \}$, where *conv* denotes the convex hull. The following cases can be considered:

Case (i) : If for all $v \in V(H_\star \cap P)$ so that $v \notin \mathrm{C}$, $g(v) \leq 0$, and v is inaccessible from x via all the possible upward paths in $\{1, \ldots, p\}$, then x can be fathomed.

Case (ii) : $z^k \in \mathrm{C}$ for all $k = 1, \ldots, p$. Then for all $z^k \neq x^\star$, $g(z^k) > 0$. Since $g(x) > 0$, any upward path, initiated at x and ending at z^k, does not intersect the boundary of g, ∂g. Thus no better solution to (LRCP) can be found on such a path.

It is easy to see that under condition (i) and (ii), x can be eliminated from further consideration and thus added to the set $\mathcal{F}$.

Since several implementations are possible, we present pseudo-code so that readers may fully understand the basic steps of this implementation.

2.3. PSEUDO-CODE

Denote by $pred\,(u)$ the most recent predecessor of the vertex u and $Fathom\ 3(x)$ is the name of a procedure which checks whether or not *fathoming criterion type three* is satisfied at vertex x.

BEGIN: [**Edge-Search Algorithm**]
$x^\circ = \text{Argmin}\,\{c^\top x \mid x \in \mathrm{P}\}$
$\hat{x} = \text{Argmax}\,\{c^\top x \mid x \in \mathrm{P}\}$
$u = x^\circ, \quad UB = c^\top \hat{x}, \quad \mathcal{F} = \phi$

While $(u \neq x^\circ \ \vee \ N(u) \neq \phi)$
  While $(N(u) \neq \phi)$! While not fathom 1
    Select $x \in N(u) \ni x \notin \mathcal{F}$
    $N(u) = N(u) \setminus \{x\}$
    if $(x \in G)$
      $\hat{\alpha} = \text{Argmin}\,\{0 \leq \alpha \leq 1 \mid g(u + \alpha\,(x - u)) = 0\}$
      $\hat{x} = u + \hat{\alpha}\,(x - u)$
      if $(c^\top \hat{x} < UB), \ UB = c^\top \hat{x}, \ x^\star = \hat{x}$ **end**
      $\mathcal{F} = \mathcal{F} \cup \{x\}$! Else fathom 2 satisfied
    else if $(c^\top x \geq UB), \ [x \notin G]$
      $\mathcal{F} = \mathcal{F} \cup \{x\}$
      else if $Fathom\ 3(x)$
        $\mathcal{F} = \mathcal{F} \cup \{x\}$! Fathom 3 satisfied
        else $[c^\top x < UB, \ x \notin G]$
          $u = x$
        end
      end
    end
  end (inner while)
  $\mathcal{F} = \mathcal{F} \cup \{u\}$
  $u = pred\,(u)$
end (outer while)

END: [**Edge-Search Algorithm**]

3. Construction of Test Problems

This section briefly discusses systematic procedures for generating test problems for LRCP's and concave minimization problems whose globally optimal solutions are known (see also [17, 20, 14, 5, 15, 16]). Subsection 3.1 briefly discusses the

random generation of the A,b,c data. Subsection 3.2 describes a modification of the Sung–Rosen method [20], for the generation of concave minimization test problems, in order to generate LRCP's with known global solutions. Subsection 3.3 presents a new method for the generation on LRCP's with known global solutions. Section 4 presents tables of our computational results using the edge search algorithm on the problems generated by the methods of this section.

3.1. Random Polytopes

In our computational experiments, we generated the polytope P and vector c randomly. We employ the method of Horst and Thoai [13], with a minor modification as follows:

Given m and n, for $i = 1, \ldots, (m-1)$ and $j = 1, \ldots, n$, the elements of the matrix A, a_{ij} and c_j are uniformly generated in the range $[-1, 1]$. The last row of A is uniformly generated in $[0, 1]$. Then, we let

$$b_i = \sum_{j=1}^{n} a_{ij} + 2u, \quad i = 1, \ldots, m, \quad u \in [0, 1].$$

For simplicity of data handling, in our computational experiments, we multiplied all the data elements by 10 and then rounded them down to the nearest integer. Then b_m is replaced with $10b_m$. It is obvious that $P = \{ x \mid Ax \leq b,\ x \geq 0 \} \neq \phi$ and is bounded.

3.2. Modified Sung–Rosen Method for LRCP

We briefly discuss a modification of the Sung and Rosen [20, 5] for the generation of LRCP test problems. Let $\bar{A} = \binom{A}{-I_n}$, $\bar{b} = \binom{b}{0}$, where I_n is an $n \times n$ identity matrix and 0 is a column vector of n zeros. In this notation, $P = \{ x \in R^n \mid \bar{A}x \leq \bar{b} \}$. Let s° be an edge point of P (and not a vertex of P). Therefore, s° is a vertex of $\Omega = \{ x \in P \mid c^\top x \leq z' \}$, where $z' = c^\top s^\circ$. Let B be the $(n-1) \times n$ submatrix of $\bar{A}$ corresponding to the $n-1$ tight constraints. Let $\tilde{B} = \binom{B}{c^\top}$, and let $\hat{b}$ be composed of the components of $\bar{b}$ that correspond to the rows of $\bar{A}$ and $\tilde{b} = \binom{\hat{b}}{z'}$. Define $f(x) = -\|\tilde{B}x - r\|^2$ (see [20] for the definition of r). Then s° is globally optimal for the concave minimization problem $\min \{ f(x) \mid x \in \Omega \}$; also, s° is globally optimal for the linear reverse convex programming problem, $\min \{ c^\top x \mid x \in P,\ g(x) \leq 0 \}$, where $g(x) = f(x) + \frac{1}{4} \|\tilde{b} - u\|^2$ (again, see [20] for the definition of u). Note that $f(s^\circ) = -\frac{1}{4} \|\tilde{b} - u\|^2$. The vector s° is chosen as follows. Let $\hat{x} = Argmax\{c^\top x | x \in P\}$ and let $x^\circ = Argmin\{c^\top x | x \in P\}$. Let x' be a randomly chosen point on the line segment $[x^{min}, x^{max}]$. Then s° is chosen so that it's one of the vertices of $P \cap \{x | c^\top x = c^\top x'\}$.

3.3. LRCP Test Problem Generation

Let $P = \{ x \mid Ax \leq b, \quad x \geq 0 \}$ be a nonempty polytope in R^n, and c be a n-vector in R^n. Also, let:

$$\begin{aligned} \hat{x} &= Argmax \{ c^\top x \mid x \in P \}, \\ x^\circ &= Argmin \{ c^\top x \mid x \in P \}. \end{aligned}$$

Let $x' \in P$ be a point on the line segment $[x^\circ, \hat{x}]$. Let $z' = c^\top x'$ and let

$$s^\circ \in Argmax \{ c^\top x \mid x \in P, \ c^\top x \leq z' \}.$$

That is, let $\Omega = P \cap \{ x \mid c^\top x \leq z' \}$ and assume that s° is a nondegenerate vertex of Ω so that $c^\top s^\circ = z'$ and so that it is located in the relative interior of an edge of P. Next, let δ^j, $j = 1, \ldots, n$ denote the normalized directions from s° to its n neighboring vertices and let

$$y^j = s^\circ + \delta^j \qquad j = 1, \ldots, n, \quad \text{and}$$

$$D = [(y^1 - s^\circ), (y^2 - s^\circ), \ldots, (y^n - s^\circ)] = [\delta^1, \delta^2, \ldots, \delta^n]$$

be a matrix whose j^{th} column is δ^j. Clearly, D^{-1} exists and the hyperplane $H' = \{ x \mid e^\top D^{-1}(x - s^\circ) = 1 \}$ passes through y^j, $j = 1, \ldots, n$, where e is an n-vector of ones. Let

$$\bar{x} = Argmax \{ e^\top D^{-1} x \mid x \in \Omega \}.$$

Consider the simplex $\mathcal{S} = conv\{ s^\circ; s^1, s^2, \ldots, s^n \}$ defined by

$$\mathcal{S} = cone\{\delta^1, \delta^2, \ldots, \delta^n\} \cap \{x | e^\top D^{-1} x \leq e^\top D^{-1} \bar{x}\}.$$

The vertices s^j, $j = 1, \ldots, n$ are found as follows: Let

$$\bar{x} = s^\circ + \alpha (y - s^\circ)$$

for some $\alpha \in R^1$ and where y is the intersection point of the line segment $[s^\circ, \bar{x}]$ with the plane H'. Then

$$\begin{aligned} (\bar{x} - s^\circ) &= \alpha (y - s^\circ) \\ e^\top D^{-1}(\bar{x} - s^\circ) &= \alpha e^\top D^{-1}(y - s^\circ) \\ &= \alpha. \end{aligned}$$

Thus, the constant α is given by

$$\alpha = e^\top D^{-1}(\bar{x} - s^\circ). \tag{2}$$

Therefore,

$$s^j = s^\circ + \alpha (y^j - s^\circ) = s^\circ + \alpha \delta^j, \quad j = 1, \ldots, n.$$

Let $S = [s^1, s^2, \ldots, s^n]$ be a matrix whose j^{th} column is s^j. Then,

$$S = s^\circ e^\top + \alpha D. \tag{3}$$

To ensure that Ω is entirely contained in S, we let

$$\alpha = e^\top D^{-1}(\bar{x} - s^\circ) + \varepsilon \tag{4}$$

for $\varepsilon > 0$ and small. Next, we construct a concave function g that passes through all vertices of the simplex, including s°, thereby forcing s° to be a global solution for

$$\begin{aligned} &\textit{Minimize} \quad c^\top x \\ &\textit{Subject to:} \quad x \in P \qquad\qquad \text{(LRCP)} \\ &\qquad\qquad g(x) \leq 0. \end{aligned} \tag{5}$$

We now describe the generation of the function g (also see [16], [14], and [17] for related work). Let $h(x)$ be an arbitrary concave function on R^n and define

$$g(x) = h(x) + d^\top x + g^\circ.$$

Since it is desired that g vanish at s°; $s^1, s^2, \ldots, s^n$, we obtain:

$$g(s^j) - g(s^\circ) = 0, \quad j = 1, \ldots, n.$$

the latter implies

$$d^\top (s^j - s^\circ) = h(s^\circ) - h(s^j), \quad j = 1, \ldots, n.$$

Thus, if B is an $n \times n$ matrix whose i^{th} row is $(s^i - s^\circ)^\top$ and β is an n-vector whose j^{th} component is given by $\beta_j = h(s^\circ) - h(s^j)$ for $i, j = 1, \ldots, n$, respectively, then

$$Bd = \beta.$$

Clearly, B is a nonsingular matrix. Therefore, we obtain

$$d = B^{-1}\beta$$

and

$$g^\circ = -h(s^\circ) - d^\top s^\circ.$$

Thus,

$$\begin{aligned} g(x) &= h(x) + d^\top x + g^\circ \\ &= h(x) + B^{-1}\beta\, x - h(s^\circ) - B^{-1}\beta\, s^\circ \\ &= h(x) + B^{-1}\beta\,(x - s^\circ) - h(s^\circ) \end{aligned} \tag{6}$$

is the desired concave function.

A variety of choices may be assumed for $h(x)$. Thus, a family of test problems with known global solutions can be generated.

In practice, in order to guarantee the presence of non-global local minima, a tighter simplex containing Ω may be found by tilting the plane H' or appending a new constraint to the generated problem. This modification usually increases the number of local solutions and does so without affecting the predetermined global solution. To this end, let $NV(s^\circ)$ denote the set of neighboring vertices of s° and let

$$j = Argmin\{\, c^\top \delta^k \mid k = 1, \ldots, n \,\}.$$

Then δ^j is replaced by $\theta \cdot \delta^j$ for some θ (for instance, $\theta = 100$). Next, let

$$k = Argmax\{\, \|x^i - s^\circ\| \mid x^i \in NV(s^\circ),\ c^\top(x^i - s^\circ) = 0 \,\}.$$

Furthermore, let $\eta^k = (x^k - s^\circ)/\|x^k - s^\circ\|$ denote the normalized direction from s° to x^k and update s^k, the k-th column of matrix S, by $s^k = x^k + \hat{\theta} \cdot \eta^k$ for some step-size $\hat{\theta}$ (for instance, $\hat{\theta} = 2.0$). Finally, add the constraint $e^\top (S - s^\circ e^\top)^{-1} x \leq 1$ to the problem. Clearly, $(S - s^\circ e^\top)^{-1}$ is easily obtained from the computed D^{-1}.

3.3.1. *Example*

The following $m = 4$, $n = 5$ problem was generated by the method of 3.3.

$$\begin{aligned}
Minimize \quad & -8x_1 + 3x_2 - 2x_3 + 4x_4 + 8x_5 \\
Subject\ to: \quad & \\
& -6x_1 + 4x_2 - 9x_4 - 10x_5 \leq -10 \\
& -9x_1 + 9x_2 + 7x_3 + x_4 - 2x_5 \leq 15 \\
& 4x_1 + 7x_2 + 6x_3 + 9x_4 + 8x_5 \leq 460 \\
& -18x_1 + 19x_2 - 39x_3 + 21x_4 - 21x_5 \leq 1000 \\
& g(x) = h_1(x) + d^\top x + g^\circ \leq 0
\end{aligned}$$

where,

$$h_1(x) = -x^\top x, \quad \text{and } x_i \geq 0, \quad \text{for } i = 1, \ldots, 5.$$

$$\begin{aligned}
g^\circ &= -2.286151585213219e+04 \\
d(1) &= 3.518274472794502e+02 \\
d(2) &= -5.987833002926723e+01 \\
d(3) &= 6.184337016912109e+01 \\
d(4) &= -8.381979490435479e+01 \\
d(5) &= -5.318398368208818e+01
\end{aligned}$$

The global solution $x = s^\circ$ and the optimal value are:

$$\begin{aligned} x(1) &= 8.120438164472780e+01 \\ x(3) &= 2.253041223684811e+01 \\ z &= -6.946958776315187e+02. \end{aligned}$$

Any component not listed is zero. The feasible soultion y given by:

$$\begin{aligned} y(1) &= 9.161896191727800e+01 \\ y(2) &= 1.336059319012680e+01 \end{aligned}$$

is a strict local solution with an objective value of $z = -692.8699157678436$. This can be verified by linearizing the function g about y, appending the linearized constraint to P, solving the resulting linear program, and observing that y uniquely solves that linear program. In fact, if one starts with the initial point $(80, 10, 0, 0, 0)$, MINOS5.3 converges to the solution, to eight decimal places,

$$\begin{aligned} y(1) &= 9.161896189e+01, \\ y(2) &= 1.336059321e+01. \end{aligned}$$

Indeed, if one starts from the initial point $(60, 0, 0, 0, 0)$, MINOS5.3 converges to another non-global local minimum

$$\begin{aligned} y(1) &= 8.600196563e+01, \\ z &= -6.88015725e+02. \end{aligned}$$

4. A Collection of Test Problems

This section presents numerical results for two types of problem construction: the method presented above in Subsection 3.3, and a modification of the Sung and Rosen method [20] for concave minimization, as discussed in Subsection 3.2.

A SUN SPARCstation 10 was used to solve the test problems given in Tables II and III. It should be emphasized that the numbers of function evaluations, and hence the CPU times, that are reported in these tables are the results of a highly accurate line-search routine. Table II contains the results for LRCP's. Problems 1–9 were generated by the methods of 3.1 and 3.2. Problem $i^\star$ differs from problem i only in that a closed form solution for each line-search was utilized (recall, the Sung–Rosen method produces a quadratic function). Problem 10 contains the network flow capacity expansion example of [4]. Problems 11–18 were generated by the method of Section 3.1 and 3.3. In addition, the functions h are h_1, $h_2, \ldots, h_5$, where

Table II. Linear reverse convex test problems

Edge Search Performance							
No.	Row/Col $m \times n$	Filename ftp site	#Pivot	#Func. Eval.	#First Points	#Fathom Type 3	Time seconds
1	5 × 10	sr5x10.dat	94	1410	112	10	.48
1*	5 × 10	sr5x10.dat	94	309	112	10	.37
2	15 × 10	sr15x10.dat	106	728	105	12	.62
3	10 × 40	sr10x40.dat	256	24654	2173	24	36.08
3*	10 × 40	sr10x40.dat	256	3470	2173	24	11.79
4	20 × 15	sr20x15.dat	340	5940	437	44	3.55
4*	20 × 15	sr20x15.dat	340	1551	437	44	2.68
5	25 × 15	sr25x15.dat	32829	124650	804	7582	344.53
6	25 × 18	sr25x18.dat	27197	115687	168	5517	243.89
7	25 × 20	sr25x20.dat	3228	22862	1170	778	35.42
8	30 × 15	sr30x15.dat	34079	111845	43	8053	316.00
9	30 × 18	sr30x18.dat	10911	82093	3037	2439	134.00
9*	30 × 18	sr30x18.dat	10911	54020	3037	2439	126.00
10	33 × 27	bj.dat	7701	22421	23	2771	57.10
11	6 × 8	jm6x8h4.dat	176	697	31	32	.39
12	11 × 5	jm11x5h1.dat	38	356	21	1	.27
13	10 × 20	jm10x20h1.dat	4761	38435	5316	1310	11.00
14	11 × 15	jm11x15h1.dat	597	3605	405	146	1.98
15	11 × 20	jm11x20h3.dat	1537	13418	2388	315	6.35
16	11 × 40	jm11x40h5.dat	34087	437275	2340	11354	106.00
17	21 × 10	jm21x10h2.dat	294	1466	142	69	2.04
18	30 × 10	jm30x10h1.dat	728	3408	349	196	5.48

Table III. Concave minimization test problems

			Edge Search Performance			
No.	Row/Col $m \times n$	Filename ftp site	#Pivot	#Func. Eval.	#First Points	Time seconds
19	3 × 5	sr3x5c.dat	10	34	6	.08
20	5 × 3	sr5x3c.dat	12	24	4	.08
21	8 × 10	sr8x10c.dat	891	1713	445	1.61
22	15 × 10	sr15x10c.dat	12421	37147	6194	36.77
23	15 × 10	2sr15x10c.dat	12171	30391	6074	38.23
24	20 × 15	sr20x15c.dat	23317	69844	11644	31.14
25	25 × 18	sr25x18c.dat	34136	85203	16998	367.3
26	25 × 20	sr25x20c.dat	34154	85234	16999	401.8
27	10 × 20	fpc.dat	34044	102048	16999	98.0
28	6 × 4	ht249c.dat	12	29	5	.08
29	9 × 5	hthc.dat	19	32	6	.17
30	21 × 50	ht21x50ch5.dat	34982	98999	17060	120.2

$$h_1(x) = -x^\top x$$

$$h_2(x) = -x^\top x + \left[\sum_{i=1}^{n} \sqrt{|x_i|}\right]^{\frac{3}{2}}$$

$$h_3(x) = -x^\top \Lambda x$$

$$h_4(x) = -(r^\top x)\, ln(1 + r^\top x), \quad r = (\tfrac{1}{1}, \tfrac{1}{2}, \tfrac{1}{3}, \cdots, \tfrac{1}{n})^\top.$$

$$h_5(x) = -(x_1 + \tfrac{1}{2}\, x_2 + \tfrac{2}{3}\, x_3 + \cdots + \tfrac{n-1}{n}\, x_n)^{\frac{3}{2}}$$

The diagonal matrix Λ in h_3 in Problem 16 is:

$$\Lambda = Diag\,[9, 4, 8, 7, 7, 9, 1, 9, 5, 8, 2, 7, 7, 8, 8, 3, 9, 9, 7, 9]$$

These functions above are taken from [6, 12].

Table IIIcontains results for concave minimization problems. Problems 19–26 were generated by the methods of Subsections 3.1 and 3.2. Problem 27 is taken from Floudas and Pardalos [7], Problem 28 is an example in Horst and Tuy [12],

and Problem 29 is taken from Horst and Thoai [13]. Problem 30 was created by the method of Subsection 3.1 and the objective function, h_5 from [12], is used.

References

1. Avriel, M. and Williams, A.C. (1970), Complementary Geometric Programming, *SIAM Journal of Applied Mathematics* **19**, 125–141.
2. Avriel, M. and Williams, A.C. (1971), An Extension of Geometric Programming with Applications in Engineering Optimization, *Journal of Engineering Mathematics* **5**, 187–194.
3. Bansal, P.P. and Jacobsen, S.E. (1975), Characterization of Basic Solutions For a Class of Nonconvex Programs, *Journal of Optimization Theory and Applications* **15**, 549–564.
4. Bansal, P.P. and Jacobsen, S.E. (1975), An Algorithm for Optimizing Network Flow Capacity under Economies-of-Scale, *Journal of Optimization Theory and Applications* **15**, 565–586.
5. BenSaad, S. and Jacobsen, S.E. (1990), A Level Set Algorithm for a Class of Reverse Convex Programs, *Annals of Operations Research* **25**, 19–42.
6. Benson, H.P. and Sayin, S. (1994), A Finite Concave Minimization Algorithm Using Branch and Bound and Neighbor Generation, *Journal of Global Optimization* **5**(1), 1–14.
7. Floudas, C.A. and Pardalos, P.M. (1990), *A Collection of Test Problems for Constrained Global Optimization Algorithms*, Springer-Verlag, Lecture Notes in Computer Sciences **455**.
8. Gurlitz, T.R. and Jacobsen, S.E. (1991), On the Use of Cuts in Reverse Convex Programs, *Journal of Optimization Theory and Application* **68** (2).
9. Hillestad, R.J. and Jacobsen, S.E. (1980), Linear Programs with an Additional Reverse-Convex Constraint, *Journal of Applied Mathematics and Optimization* **6**, 257–269.
10. Hillestad, R.J. and Jacobsen, S.E. (1980), Reverse-Convex Programming, *Journal of Applied Mathematics and Optimization* **6**, 63–78.
11. Hillestad, R.J. (1975), Optimization Problems Subject to a Budget Constraint with Economies of Scale, *Operations Research, Journal of Applied Mathematics and Optimization* **23** (6), 1091–1098.
12. Horst, R. and Tuy, H. (1990), *Global Optimization: Deterministic Approaches*, Springer-Verlag, Berlin-New York.
13. Horst, R. and Thoai, N.V. (1989), Modification, Implementation and Comparison of Three Algorithms for Globally Solving Linearly Constrained Concave Minimization Problems, *Computing* **42**, 271–289.
14. Kalantari, B. and Rosen, J.B. (1986), Construction of Large-Scale Global Minimum Concave Quadratic Test Problems, *Journal of Optimization Theory and Applications* **48**, 303–313.
15. Moshirvaziri, K. (1994), A Generalization of the Construction of Test Problems for Nonconvex Optimization, *Journal of Global Optimization* **5**(1), 21–34.
16. Moshirvaziri, K. (1994), Construction of Test Problems for a Class of Reverse Convex Programs, *Journal of Optimization Theory and Applications* **81**(2), 343–354.
17. Pardalos, Panos M. (1987), Generation of Large-Scale Quadratic Programs for Use as Global Optimization Test Problems, *ACM Transaction on Mathematical Software* **13** (2), 143–147.
18. Rosen, J.B. (1966), Iterative Solution of Non-linear Optimal Control Problems, *SIAM Journal of Control* **4**, 223–244.
19. Rosen, J.B. (1983), Global Minimization of a Linearly Constrained Concave Function by Partition of Feasible Domain, *Mathematics of Operations Research* **8**, 215–230.
20. Sung, Y. and Rosen, J.B. (1982), Global Minimum Test Problem Construction, *Mathematical Programming* **24**, 353–355.
21. Thuong, Nguyen Van and Tuy, Hoang (1985), *A Finite Algorithm for Solving Linear Programs with an Additional Reverse Convex Constraint*, Springer-Verlag, Lecture Notes in Economics and Mathematical Systems **225**, 291–302.
22. Tuy, Hoang (1987), Convex Programs with an Additional Reverse Convex Constraint, *Journal of Optimization Theory and Applications* **52**(3), 463–485.

23. Tuy, Hoang (1985), A General Deterministic Approach to Global Optimization via D.C. Programming, In: Hiriart-Urruty, J.B. (ed.), *Fermat Days: Mathematics for Optimization*, Elsevier, Amsterdam, 137–162.
24. Ueing, U. (1972), A Combinatorial Method to Compute a Global Solution of Certain Non-convex Optimization Problems, in *Numerical Methods for Nonlinear Optimization*, F.A. Lootsma (ed.), Academic Press, 223–230.