A LADDER METHOD
FOR LINEAR SEMI-INFINITE PROGRAMMING

YANQUN LIU AND MING-FANG DING

School of Mathematical and Geospatial Sciences
RMIT University
GPO Box 2476, Melbourne
Victoria 3001, Australia

Abstract. This paper presents a new method for linear semi-infinite programming. With the introduction of the so-called generalized ladder point, a ladder method for linear semi-infinite programming is developed. This work includes the generalization of the inclusive cone version of the fundamental theorem of linear programming and the extension of a linear programming ladder algorithm. The extended ladder algorithm finds a generalized ladder point optimal solution of the linear semi-infinite programming problem, which is approximated by a sequence of ladder points. Simple convergence properties are provided. The algorithm is tested by solving a number of linear semi-infinite programming examples. These numerical results indicate that the algorithm is very efficient when compared with other methods.

1. Introduction. Linear semi-infinite programming (LSIP) has been an attractive research area of optimization for decades. Several well-known numerical methods ([2, 5, 6, 7, 9]) have been developed over the years. Examples are the discretization methods ([8, 16]), cutting plane methods ([2, 3, 18]), exchange methods ([6, 18]), dual parametrization methods ([10, 15]), interior point methods ([4, 11]), and local reduction methods ([7, 9]). More detailed list of references can be found in [2, 5, 6, 7] and the references cited therein. Worth mentioning is the more comparable simplex based exchange algorithms developed or documented in [1, 2, 5] and [6].

Successful LSIP methods have to be able to deal with the infinitely many constraints efficiently and should avoid common drawbacks that cause numerical difficulties and slow convergence. These common drawbacks include increasingly large subproblem sizes (e.g. cutting plane, discretization), non-linear subproblems (e.g. local reduction, dual parameterization), and feasibility check over densely sampled index points. Many improvements of various semi-infinite programming (SIP) methods have been made by reducing the subproblem sizes, relaxing the need for a global optimal solution when non-linear subproblem occurs, or reducing the number of iterations to speed up the convergence. The exchange methods, the relaxed cutting plane methods ([18]), and the accelerated central cutting plane method ([2]) are examples of efforts to improve numerical performance.

In this paper, we present a new approach to LSIP that is obtained by generalizing a linear programming (LP) method ([12]). We first provide an optimality result for...
LSIP, which can be considered as an LSIP extension of the inclusive cone version of the fundamental theorem for linear programming. The idea of this optimality result, based on the notion of inclusive cone, naturally leads to the so-called centered climbing algorithm. This algorithm works in the following way. At the beginning of the algorithm, an initial artificial ladder is obtained or constructed. Starting from this initial ladder, a sequence of ladders is constructed iteratively, giving rise to a sequence of approximate solutions which converges to a special optimal solution – a feasible generalized ladder point.

Before giving the problem formulation, we first set up some notational conventions of representing matrices, including vectors in Euclidean spaces. In case the contents of a matrix need to be displayed, we will often present them in row form, with elements in the same row separated by commas and different rows separated by semicolons. For example, \( A = [a, b; c, d] \) represents the \( 2 \times 2 \) matrix whose first row elements are \( a \) and \( b \) and second row elements are \( c \) and \( d \), in their given orders. If \( a_1, a_2, \ldots, a_m \) are \( n \)-dimensional row vectors and \( b_1, b_2, \ldots, b_n \) are \( m \)-dimensional column vectors, then \( A = [a_1; a_2; \ldots; a_m] \) and \( B = [b_1, b_2, \ldots, b_n] \) are both \( m \times n \) matrices. Furthermore, with this notation, the elements of row and column vectors, say \( a_1 \) and \( b_1 \) above, can be conveniently displayed as \( a_1 = [a_{11}, a_{12}, \ldots, a_{1n}] \) and \( b_1 = [b_{11}; b_{12}; \ldots; b_{m1}] \). We will use \( \mathbb{R}^n \) and \( \mathbb{R}_n \) to represent the \( n \)-column and the \( n \)-row vector spaces, respectively.

In this paper, we consider the following LSIP problem.

**Problem P.**

\[
\begin{align*}
\text{min} & \quad f(x) = c^T x \tag{1} \\
\text{s. t.} & \quad a(t)x - b(t) \leq 0 \quad \text{for} \quad t \in [\alpha, \beta] \subset \mathbb{R} \tag{2}
\end{align*}
\]

where \( x = [x_1; x_2; \ldots; x_n] \in \mathbb{R}^n \) is the decision vector, \( c = [c_1; c_2; \ldots; c_n] \in \mathbb{R}^n \) is a constant vector, and \( a(t) = [a_1(t), a_2(t), \ldots, a_n(t)] : t \to \mathbb{R}_n \) is a row vector function and \( b(t) : t \to \mathbb{R} \) is a scalar function.

**Assumption 1.** In problem P, both constraint functions \( a(t) \) and \( b(t) \) are continuously differentiable on \([\alpha, \beta]\) and \( a(t) \neq 0 \) for all \( t \in [\alpha, \beta] \).

**Assumption 2.** There exist \( s_1, s_2, \ldots, s_n \in [\alpha, \beta] \) such that the vectors \( a(s_1), a(s_2), \ldots, a(s_n) \) form a basis for \( \mathbb{R}_n \).

**Assumption 3.** If the feasible region of problem P is non-empty, it contains a Slater point.

The outline of this paper is as follows. Following this introductory section, in Section 2, the concepts of inclusive cone \((\text{[12]})\) and generalized ladder point are introduced for linear semi-infinite programming problems. An optimality result is given in terms of generalized ladder point. In Section 3, a technique for finding an initial artificial ladder is first discussed. This is followed by the main algorithm and a number of comments on some details of its iterative steps. At the end of this section a simple convergence result is presented. Before a concluding summary that ends the paper, the efficiency and performance of the algorithm are demonstrated by solving a number of numerical examples in Section 4.

2. **An optimality result.** For linear programming problems, it is shown in \([12, 13]\) that the notion of inclusive cone can play an important part in developing the classical linear programming theory including the optimality and the duality
results. It can also be used to develop a class of algorithms for solving LP problems numerically, namely the ladder algorithms ([12]). Similar ideas apply to the LSIP case. We present an optimality result in this section.

Note that the inclusive cone concept applies directly to problem P. The cone generated by \( n \) linearly independent normal vectors \( a^T(t_k), k = 1, 2, \ldots, n \), is called an inclusive cone if it contains the vector \(-c\), where \( t_k \in [\alpha, \beta] \) for \( k = 1, 2, \ldots, n \). If the cone generated by \( a^T(t_k), k = 1, 2, \ldots, n \), is an inclusive cone we call the index set \( I = \{t_1, t_2, \ldots, t_n\} \) the generator of the inclusive cone and the region \( L(I) = \{x \in \mathbb{R}^n \mid a(t_k)x \leq b(t_k), k = 1, 2, \ldots, n\} \) the associated inclusive region or ladder. The index set \( I = \{t_1, t_2, \ldots, t_n\} \) is also referred to as the generator of \( L(I) \). The unique solution of the linear system \( a(t_k)x = b(t_k), k = 1, 2, \ldots, n \), denoted by \( x_I \), is called the ladder point of \( L(I) \). Since the feasible region of an LSIP problem is in general no longer a polytope, inclusive cone based linear programming results may not automatically apply to LSIP. For a proper generalization, we extend the concept of ladder point in the following.

**Definition 2.1.** A point \( x \in \mathbb{R}^n \) is called a generalized ladder point of problem P if there is a sequence of ladder points \( x^i, i = 1, 2, \ldots \), with corresponding sequence of ladders \( L(I_i) \) such that \( x^i \to x \) (\( i \to \infty \)).

It is clear that ladder points are special generalized ladder points. In the ladder method for linear programming, ladder points play an important role. The ladder algorithms are designed to obtain an optimal ladder solution, whose existence in case an optimal solution exists is guaranteed by the following ladder point version of the fundamental theorem of linear programming.

**Lemma 2.2.** ([12]) For an arbitrary inequality constrained \( n \)-dimensional linear programming problem whose constraint coefficient matrix has rank \( n \), the following statements hold:

1. If the problem has no optimal solution, then it is either infeasible or unbounded.
2. If the problem has an optimal solution, it must have one at a ladder point.

For LSIP problems, feasible region may no longer contain a ladder point. However, it contains a generalized ladder point if an optimal solution exists, as the following result shows.

**Theorem 2.3.** Consider problem P specified in Section 1. The following statements are true.

(a) A generalized ladder point is an optimal solution if and only if it is feasible.
(b) Let Assumptions 1 - 3 be satisfied. Problem P has an optimal solution if and only if it has one at a feasible generalized ladder point.
(c) Let Assumptions 1 - 3 be satisfied. Problem P has no optimal solution if and only if it either has no generalized ladder point or all its generalized ladder points are infeasible.

**Proof.** (a) This statement is obviously true since the objective value at a generalized ladder point is a lower bound of the optimal value, and hence a feasible generalized ladder point must be an optimal solution.

(b) We need only to prove the necessity. Assume that \( x^* \) is an optimal solution of problem P. Since the feasible region of problem P contains a Slater point, as a direct consequence of the KKT conditions (see e.g. Theorem 6 in [10]), there are
\[ t_1, t_2, \cdots, t_k \in [\alpha, \beta] \text{ for some } k, 1 \leq k \leq n, \text{ such that } a(t_i), i = 1, 2, \cdots, k \text{ are linearly independent, } \]
\[ a(t_i)x^* = b(t_i), \; i = 1, 2, \cdots, k, \quad (3) \]

and
\[ -c = \sum_{i=1}^{k} \lambda_i a^T(t_i), \quad \lambda_i > 0, \; i = 1, 2, \cdots, k. \quad (4) \]

According to Assumption 2, we can expand \( \{a(t_1), \cdots, a(t_k)\} \) to \( \{a(t_1), \cdots, a(t_n)\} \) to form a basis for \( R_n \). Obviously, \( I = \{t_1, t_2, \cdots, t_n\} \) generates a ladder \( L(I) \). We consider two cases.

**Case 1.** Problem \( P \) has a non-degenerate ladder \( L(s_1, s_2, \cdots, s_n) \). That is, none of the edges of \( L(s_1, s_2, \cdots, s_n) \) is orthogonal to \( c \). Let \( B^k = \{\alpha + \frac{1}{k}(\beta - \alpha) | i = 0, 1, \cdots, 2^k\} \cup \{s_1, s_2, \cdots, s_n\} \). Then, the following problem

**Problem \( P_{B^k} \).**

\[
\min \quad c^T x \\
\text{s. t.} \quad a(t)x \leq b(t) \text{ for } t \in B^k
\]

has an optimal solution, denoted by \( x^k \), for all \( k = 1, 2, \cdots \), as it is feasible and has a ladder. Clearly,
\[ x^k \in X \ni \{x | x \in L(I) \text{ and } c^T x \leq c^T x^*\}. \]

Note that the set \( X \) in the above expression is compact, the sequence \( \{x^k\} \) has a converging subsequence \( \{x^{k_j}\}_{j=1}^{\infty} \). Let \( x^{k_j} \rightarrow \bar{x} \), then \( \bar{x} \) satisfies \( a(t)\bar{x} \leq b(t) \) for all \( t \in \bigcup_{k=1}^{\infty} B^k \) and hence for all \( t \in [\alpha, \beta] \). We have \( c^T \bar{x} = c^T x^* \). Therefore, \( \bar{x} \), as a generalized ladder point, is an optimal solution.

**Case 2.** All ladders of problem \( P \) are, especially \( L(I) \), is degenerate. Since \( t_1, t_2, \cdots, t_k \) satisfy \( (3) \) and \( (4) \), the ladder point \( x^0 \) of \( L(I) \) satisfies \( c^T x^0 = c^T x^* \). We consider the linear system

\[ a(t_i)x = b(t_i), \; i = 1, 2, \cdots, k \quad (5) \]

As \( a(t), \; i = 1, 2, \cdots, k \), are linearly independent, by solving system \( (5) \) we can represent \( k \) variables of \( x \) in terms of its remaining \( n - k \) variables which are denoted by \( y = [y_1; y_2; \cdots; y_{n-k}] \). Thus, we have
\[ x = Gy + d \]

for some \( n \times (n-k) \) matrix \( G \) and vector \( d \in R^n \).

Consider the following problem.

**Problem \( \bar{P} \).**

\[
\min \quad \bar{c}^T y \\
\text{s. t.} \quad \bar{a}(t)y \leq \bar{b}(t) \text{ for } t \in [\alpha, \beta]
\]

where
\[ \bar{a}(t) = a(t)G, \]
\[ \bar{c} = -\bar{a}^T(t_{k+1}) - \bar{a}^T(t_{k+2}) - \cdots - \bar{a}^T(t_n), \]
and
\[ \bar{b}(t) = b(t) - a(t)d. \]

It is obvious that each feasible point \( y \) of problem \( \bar{P} \) corresponds to the optimal solution \( x = Gy + d \) of problem \( P \), and each ladder \( L(s_1, s_2, \cdots, s_{n-k}) \) with ladder
point $y^0$ of problem $\bar{P}$ corresponds to the ladder $L(t_1, t_2, \cdots, t_k, s_1, s_2, \cdots, s_{n-k})$ of problem $P$ with ladder point $x^0 = Gy^0 + d$.

With $\bar{c}$ designed above, we see that $I = \{t_{k+1}, t_{k+2}, \cdots, t_n\} \subset I$ generates a non-degenerate ladder for problem $P$. According to Case 1, problem $P$ has an optimal solution $y^*$ which is also a generalized ladder point of problem $P$. We denote by $L(s_1, s_2, \cdots, s_{n-k})$, $i = 1, 2, \cdots$ the sequence of ladders of problem $P$ whose corresponding ladder point sequence $\{y^i\}$ converges to $y^*$. Then by the above discussion, $x^* = Gy^* + d$ is an optimal solution of problem $P$ and $x^i = Gy^i + d$, $i = 1, 2, \cdots$ is a sequence of ladder points that satisfies

$$\lim_{i \to \infty} x^i = \lim_{i \to \infty} (Gy^i + d) = Gy^* + d = x^*$$

which means that $x^*$ is a generalized ladder point of problem $P$ as well.

(c) This is a direct consequence of (b). \qed

We note that in the proof of (b) and (c) above, we need only the continuity of $a(t)$ and $b(t)$. Their continuous differentiability is needed only by the algorithm in the next section.

3. Algorithm. In this section, we propose an extended ladder algorithm for solving problem $P$. The main idea is to obtain a sequence of ladders $\{L(I_i)\}$ whose corresponding sequence of ladder points, $\{x^i\}$, converges to a generalized ladder point $x^*$ which, according to Theorem 2.3, must be an optimal solution. At the start of the solution process, an initial ladder $L(I_1)$ with ladder point $x^1$ is artificially constructed. Then, a sequence of ladders $L(I_2)$, $L(I_3)$, $\cdots$, with corresponding ladder points $x^2$, $x^3$, $\cdots$, is iteratively produced in such a way that $L(I_{n+1})$ is obtained from $L(I_n)$ by climbing up (toward the feasible region) from $x^i$ along the 'centerline' of $L(I_i)$. The climbing procedure resembles the centered climbing ladder algorithm for LP ([12]).

3.1. Artificial initial ladder. In order to start the algorithm, we need an initial ladder. It is natural to try to obtain an initial ladder from $\{a(t_i) | i = 1, 2, \cdots, N\}$ where $t_i \in T_N = \{\alpha + (i/N)(\beta - \alpha) | i = 0, 1, 2, \cdots, N\} \subset [\alpha, \beta]$. In fact, in discretization methods (see e.g. [8, 16]) authors assume that when $N$ is large, the discretized problem

Problem $P_{T_N}$.

$$\min f(x) = c^T x$$

s. t. \hspace{1cm} a(t)x - b(t) \leq 0 \hspace{1cm} \text{for} \hspace{1cm} t \in T_N$$

has an optimal solution, which implies according to Lemma 2.2 that problem $P_{T_N}$ has a ladder, and hence we can select $n$ vectors from $\{a(t_i) | i = 1, 2, \cdots, N\}$. This assumption is actually stronger than expected and can be easily violated as the following example indicates.

Example 1. Consider the two-dimensional problem given by

$$\min y$$

s. t. \hspace{1cm} x \sin((t - \sqrt{\pi/2})^2) - y \cos((t - \sqrt{\pi/2})^2) \leq 1 \hspace{1cm} \text{for} \hspace{1cm} 0 \leq t \leq 2$$

For arbitrarily large $N$, the discretized problem $P_{T_N}$ is unbounded as it doesn’t include the constraint at $t = \sqrt{\pi/2}$, and hence it does not have an optimal solution, even though the original problem has infinitely many solutions. Thus, from [13],
P_{T_N} does not have a ladder and it is impossible to obtain an initial ladder from \{a(t) \mid t \in T_N\}.

In the following, instead of obtaining an initial ladder for the problem by discretization, we construct an artificial ladder that works for general problems.

Consider problem \( P \). It is clear that, if \( M \) is a sufficiently large number, by adding constraint \( c^T x \geq -M \) to problem \( P \) neither the problem’s optimal solutions if they exist nor its solvability if it is bounded or infeasible will be changed. From Assumption 2, we can expand the set \( \{-c\} \) into a basis \( \{-c, a^T(s_{i_1}), a^T(s_{i_2}), \ldots, a^T(s_{i_{n-1}})\} \) for \( R^n \). These vectors clearly generate an inclusive cone and hence generate a ladder with the corresponding ladder point given by

\[
x^0 = \left[ \begin{array}{c}
-c^T \\
a(s_{i_1}) \\
\vdots \\
a(s_{i_{n-1}})
\end{array} \right]^{-1} \left[ \begin{array}{c}
M \\
b(s_{i_1}) \\
\vdots \\
b(s_{i_{n-1}})
\end{array} \right]
\]

(10)

For notational reasons, we define \( a(\alpha - 1) = -c, b(\alpha - 1) = M \), and \( T = [\alpha, \beta] \cup \{\alpha - 1\} \). Then, we can conveniently present any ladder generator, including that of the artificial ladder, in the form of \( I \). The centerline of \( L(I_k) \) is defined as the line passing \( x^k \) and parallel to \( v^k \):

\[
l(x^k, v^k) : \quad x = x^k + tv^k, \quad -\infty < t < \infty.
\]

If \( x^k \) is not optimal, the current ladder needs to be updated and a new ladder point is obtained. For LP problems, in order to update \( L(I_k) \) to \( L(I_{k+1}) \) and \( x^k \) to \( x^{k+1} \), we determine a ‘pick’ (an entering constraint index) which corresponds to the constraint that is violated most along the centerline (12). For LSIP problems, the most violated constraint index \( t^k \in [\alpha, \beta] \) along the centerline is the global minimizer to the problem.

**Problem \( P(I_k) \).**

\[
\begin{align*}
\min_t & \quad g(t, I_k) \\
\text{s. t.} & \quad \alpha \leq t \leq \beta
\end{align*}
\]

where

\[
g(t, I_k) = \frac{b(t) - a(t)x^k}{|a(t)v^k|}.
\]

Geometrically, if the global minimizer \( t^k \) above exists, then \( a(t^k)x \leq b(t^k) \) is the constraint that is violated most at \( x^k \) along the centerline of \( L(I_k) \) (see Figure 1). However, problem \( P(I_k) \) may be not well posed, as in its objective function \( a(t)v^k \) may take zero value at some \( t \) positions. For these reasons, we modify the definition of \( g(t, I_k) \) and let

\[
g(t, I_k, \tau) = \frac{b(t) - a(t)x^k}{\sqrt{\tau + (1 - \tau)|a(t)v^k|^2}}, \quad \text{for } 0 < \tau \leq 1.
\]

(11)
We note that if the parameter $\tau = 0$, we obtain the objective function in problem $P(I_k)$. If $0 < \tau \leq 1$, $g(t, I_k, \tau)$ is well defined and is continuously differentiable on $[\alpha, \beta]$, and hence global minimum for the following modified problem $P(I_k, \tau)$ always exists.

**Problem $P(I_k, \tau)$**.

$$
\begin{align*}
\min_t & \quad g(t, I_k, \tau) \\
\text{s. t.} & \quad \alpha \leq t \leq \beta
\end{align*}
$$

Let $t^k$ be the global optimizer of problem $P(I_k, \tau)$ ($t^k$ will be called a ‘pick’ to be consistent with [12]). Then, if $\tau = 1$, the entering constraint $a(t^k)x \leq b(t^k)$ is the constraint that is violated most at $x^k$ (see Figure 1). Note that different $\tau$ values may lead to different entering constraint, and hence different ladder point sequences. We may naturally expect that some $\tau$ values are better than others. Our experiments show that, for the problems solved in Section 4, the algorithm performs much better for $0.9 \leq \tau \leq 1$ than for $0 < \tau \leq 0.2$. In general, the best $\tau$ value seems to be problem-dependent, and can be anywhere in $(0, 1]$.

Solving global optimization problems is in general a very demanding task. However, for one-dimensional smooth optimization problems, there are reliable and efficient numerical methods available. In this paper, we choose to use the bridging method given in [14] and we always start the bridging process from $t = \alpha$ (right bridging). In short, the right bridging method minimizes a bridged function initially constructed at the left end of the interval. By minimizing the bridged function, points that are not better than the current solution are skipped and a local minimum, better than the starting point, is found. Then, by updating the bridged function at the local minimum, better local minimum solutions are located. This process continues until a global optimal solution is found. For details of the bridging method, we refer interested readers to [14]. After choosing a pick (the global minimizer $t^k$), obtaining the updated ladder is the same as in the linear programming case [12]. For convenience, when the ladders are updated by choosing a pick as the global minimizer of problem $P(I_k, \tau)$ with $0 < \tau \leq 1$, we call the updating method a centered climbing method.

**Figure 1. Finding a pick along the centerline.**
A Centered Climbing Ladder Algorithm

**Step 0. Initializing**
0.1 Predetermine the value of $\tau \in (0, 1]$. Choose a small number $\varepsilon > 0$.
0.2 Find an artificial initial ladder $L(I_0)$ with generator $I_0 = \{t^0_1, t^0_2, \ldots, t^0_n\}$ and ladder point $x^0$. Set $k = 0$.

**Step 1. Checking optimality**
1.1 Solve problem $P(I_k, \tau)$ for a global minimizer $t^k$ with global optimal value $V_k$.
1.2 If $V_k > -\varepsilon$, output $x^* = x^k$.
   $\circ$ If $a(t^k)c < 0$, the problem is infeasible. Stop
   $\circ$ Else if $V^* = c^T x^* = -M$, the problem is unbounded. Stop
   $\circ$ Else, $x^*$ is taken as an optimal solution.
1.3 Otherwise, go to Step 2.

**Step 2. Updating**
2.1 Otherwise, update $L(I_k)$ with $t^k$ entering $I_k$:
   $\circ$ Use $t^k$ to replace one of $I_k$’s elements, say $t^k_{i_1}$, so that a new ladder generator $I_{k+1}$ is produced. Let $I_{k+1} = \{t^1_{i_1}, t^1_{2}, \ldots, t^k_{n}\}$, where
   $$ t_i^{k+1} = \begin{cases} \begin{align*} t_i^k, & \text{if } i \neq i_1 \\ t^k, & \text{if } i = i_1 \end{align*} \end{cases} $$
   $\circ$ Calculate the updated ladder point according to
   $$ x^{k+1} = [a(t^1_{i_1}); a(t^2_{i_1}); \ldots ; a(t^k_{n})]^{-1} [b(t^1_{i_1}); b(t^2_{i_1}); \ldots ; b(t^k_{n})]. $$
2.2 Set $k = k + 1$, go to Step 1.

The following are some comments about the above centered climbing algorithm.

(a) In the initialization step, very often we can choose $t^0_i = \alpha - 1$ and choose $t^0_i \in [\alpha, \beta]$ randomly for $i = 2, 3, \ldots, n$ to obtain an initial ladder. This way of finding an initial ladder, though not rigorous, is very efficient in many cases. A more formal but less efficient method for finding an initial ladder is the following procedure.

- First find $n$ linearly independent constraint normal vectors $a(t^0_i)$, $i = 1, 2, \ldots, n$ where $t^0_i \in [\alpha, \beta]$.
- Let $[\lambda_1; \lambda_2; \ldots ; \lambda_n] = -[a^T(t^0_i), a^T(t^0_2), \ldots , a^T(t^0_n)]^{-1}c$.
- If $\lambda_i \geq 0$ for $i = 1, 2, \ldots , n$, set $I_0 = \{t^0_1, t^0_2, \ldots, t^0_n\}$ and $I_0$ generates a ladder $L(I_0)$.
- Otherwise, identify any $\lambda_i \neq 0$. Set $t^0_i = \alpha - 1$ (note that $a(\alpha - 1) = -c^T$), then $a(t^0_i)$, $i = 1, 2, \ldots, n$ are linearly independent. And hence, $I_0 = \{t^0_1, t^0_2, \ldots, t^0_n\}$ generates a ladder $L(I_0)$.

Once $L(I_0)$ is obtained, the corresponding ladder point $x^0$ is calculated by solving the corresponding system of linear equations.

(b) The iterative process of the algorithm (Steps 1 and 2) includes some termination criteria and an updating step. The main computational task is to find a global minimum solution of the function $g(t, I_k, \tau)$ over the interval $[\alpha, \beta]$ in Step 1. As we have seen earlier, the global minimum value $V_k$ of $g(t, I_k, \tau)$ over the interval $[\alpha, \beta]$ is the negative of the ‘maximum violation’ of all the constraints at the current ladder point $x^k$ along the parameterized centerline. If the maximum constraint violation $-V_k$ is less than a prescribed accuracy $\varepsilon > 0$, the current ladder point $x^k$ is taken as an approximate optimal solution of problem P. Otherwise, the global minimizer $t^k$ of problem $P(I_k, \tau)$
determines the constraint that enters the current ladder. The current ladder is then updated and the new ladder point calculated with ease according to standard formulations (see [12]).

(c) The main advantage of the algorithm is threefold. Firstly, each entering constraint leads to an, generally speaking, effective improvement of the current solution. Though locating the constraint needs a global minimization, the bridging method is very efficient in doing this job as the computation effort involved is comparable to a few line searches. Secondly, the LSIP is treated like an LP and the updating, aside from the global minimization, is similar in complexity to a simplex step with fixed dimension. Thirdly, feasibility information can be drawn from the global optimization, and there is no need for extra feasibility check.

Before ending this section, we provide some simple facts regarding the convergence of the algorithm.

**Theorem 3.1.** Let \( \{x^k\} \) be the sequence of ladder points produced by the above algorithm. The following are true.

(a) Upon the termination of the algorithm, the output \( x^* \) is either an optimal solution, or the problem is infeasible or unbounded.

(b) If \( \{x^k\} \) is an infinite sequence, then the following hold true.

1. The value sequence \( \{c^T x^k\} \) is non-decreasing. If it diverges, i.e., \( c^T x^k \to \infty \) \( (k \to \infty) \), then problem \( P \) is infeasible.

2. Suppose, in the artificial constraint \( -c^T x \leq M \), \( M \) is sufficiently large. If the optimal solution \( x^* \) is found, but the optimal value \( -c^T x^* = M \), then problem \( P \) is unbounded.

3. Assume that \( c^T x^k \to \bar{V} \), and \( \max_{t \in T} (a(t)x^k - b(t)) \to 0 \) \( (k \to \infty) \). Then any converging subsequence of \( \{x^k\} \) converges to an optimal solution of problem \( P \).

**Proof.** Statements (a), (b.1) and (b.2) are obvious. Statement (b.3) is true because in this situation any limiting point will be a feasible generalized ladder point, and hence is optimal according to Theorem 2.3.

4. **Numerical examples.** In this section, we demonstrate the efficiency of the algorithm by solving three approximation problems. The first example is the one-sided \( L_1 \) approximation of the tangent function over \([0, 1]\) by polynomials with limited order. This is a standard LSIP test problem. The purpose of this example is to compare the new algorithm with other algorithms. As this problem becomes very sensitive when the order of the approximating polynomial is greater than 4, the algorithm seems only to be able to cope with cases where the limiting order of the polynomial is within 20 to obtain solutions that have small feasibility and objective errors. To demonstrate the algorithm’s ability of dealing with problems of larger size, two more examples are solved. They are the one-sided approximations of a given smooth function by a sine series and a cosine series of given orders, for which the authors are unaware of their appearance in any published work.

**Example 2.** Find the best \( L_1 \) approximation to the tangent function from above over all polynomial functions \( p(t) \) of order \( \leq n - 1 \) on the interval \([0, 1]\):

\[
\inf_p \left\{ \int_0^1 |p(t) - \tan(t)| dt \mid \begin{array}{c}
p \text{ is a polynomial of order } < n \\
p(t) \geq \tan(t) \text{ for all } t \in [0, 1] \end{array} \right\}
\]
where for each 

The problem can be formulated as the following LSIP problem.

\[
\begin{array}{cccccc}
\text{Table 1. Results for Example 2 with } & b(t) = \tan(t), & 0 \leq t \leq 1 \\
\text{and parameters } M = 50, & \varepsilon = 1.0\varepsilon - 7 & \\
\hline
n & \tau & \text{iteration} & \text{CPU} & \text{max} & \text{optimal} \\
& & \text{number} & \text{time} & \text{constr.} & \text{distance} \\
& & & (\text{seconds}) & \text{violation} & \|p - \tan(c)\|_1 \\
6 & 0.01 & 18 & 0.658005 & 3.026419e-15 & 4.586812e-4 \\
& 1 & 15 & 0.536005 & 1.268621e-15 & 4.586810e-4 \\
& 0.95 & 16 & 0.682806 & 2.344241e-16 & 4.586811e-4 \\
8 & 0.01 & 20 & 1.135620 & 2.178005e-14 & 2.675330e-5 \\
& 1 & 19 & 0.995219 & 0 & 2.675325e-5 \\
& 0.95 & 20 & 0.932818 & 1.955741e-16 & 2.675318e-5 \\
10 & 0.01 & 27 & 1.754113 & 0 & 1.589166e-6 \\
& 1 & 21 & 1.021227 & 3.851927e-17 & 1.589145e-6 \\
& 0.95 & 23 & 1.008900 & 4.058102e-16 & 1.589148e-6 \\
11 & 0.01 & 31 & 2.124825 & 8.627571e-14 & 3.787589e-7 \\
& 1 & 19 & 1.009225 & 5.048957e-14 & 3.737784e-7 \\
& 0.95 & 20 & 0.946825 & 9.700573e-15 & 3.742882e-7 \\
12 & 0.01 & 33 & 2.207455 & 6.047939e-14 & 1.145817e-7 \\
& 1 & 20 & 1.753125 & 2.354116e-13 & 1.112370e-7 \\
& 0.95 & 23 & 1.807025 & 2.438008e-14 & 1.123466e-7 \\
13 & 0.01 & 35 & 2.956250 & 2.051373e-10 & 4.049794e-7 \\
& 1 & 20 & 1.725000 & 4.913628e-10 & 1.590831e-7 \\
& 0.95 & 18 & 1.523750 & 3.237483e-10 & 8.011145e-8 \\
14 & 0.01 & 39 & 3.268125 & 3.202518e-10 & 9.787721e-7 \\
& 1 & 32 & 2.843750 & 1.839627e-10 & 6.052919e-7 \\
& 0.95 & 22 & 1.671650 & 7.038231e-10 & 7.424762e-8 \\
15 & 0.01 & 41 & 3.687751 & 2.893904e-10 & 2.745738e-7 \\
& 1 & 28 & 2.675305 & 8.132726e-10 & 4.655475e-7 \\
& 0.95 & 30 & 2.031225 & 6.046296e-10 & 2.924098e-7 \\
16 & 0.01 & 45 & 4.076100 & 7.884831e-10 & 8.275940e-7 \\
& 1 & 36 & 3.425000 & 8.493327e-10 & 2.667424e-7 \\
& 0.95 & 38 & 3.121875 & 8.174934e-10 & 1.616550e-7 \\
17 & 0.01 & 37 & 2.500000 & 7.549062e-8 & 4.366051e-6 \\
& 1 & 30 & 2.458435 & 7.482757e-8 & 5.687086e-6 \\
& 0.95 & 31 & 2.346875 & 2.127828e-8 & 1.464241e-6 \\
18 & 0.01 & 45 & 3.062500 & 7.048341e-8 & 7.458815e-5 \\
& 1 & 52 & 3.578125 & 9.554832e-8 & 1.977110e-4 \\
& 0.95 & 40 & 3.250000 & 7.360533e-8 & 5.147018e-5 \\
\end{array}
\]

The problem can be formulated as the following LSIP problem.

**Problem E₂.**

\[
\begin{align*}
\min & \quad c^T x \\
\text{s. t.} & \quad a(t)x \leq b(t), \quad \text{for } t \in [0, 1]\n\end{align*}
\]

where \( c = [1; \frac{1}{2}; \cdots; \frac{1}{n}] \), \( a(t) = -[1, t, \cdots, t^{n-1}] \) and \( b(t) = -\tan(t) \).

This problem has been solved by many authors (see [5], Chapter 11). It is known that for each \( n \), problem E₂ has a unique optimal solution. But, for \( n \geq 3 \) the
boundary of its feasible set becomes very flat in the $-c$ direction near the optimal solution, and for $n \geq 6$ the numerical solution become extremely sensitive ([17]).

The algorithm is tested for cases $1 \leq n \leq 18$. The computational results for $n = 6$, $n = 8$, and $10 \leq n \leq 18$ are listed in Table 1 with different choices of the centerline parameter $\tau$. The number of iterations and the CPU time listed depend not only on the problem dimension, but also the initial ladder that is constructed randomly in the initial step. Thus, these data may experience noticeable changes between different runs. The listed data in each case is from a particular run. We list randomly in the initial step. Thus, these data may experience noticeable changes not only on the problem dimension, but also the initial ladder that is constructed.

\[
\tau = 2 \cdot 10^{-5}
\]

The optimal value $c^T x^*$ can be obtained from $\|p^* - b\|_1$ according to $c^T x^* = \|p^* - b\|_1 - \log(\cos(1))$. For $1 \leq n \leq 16$, our computational results are very accurate in terms of both feasibility and objective optimality. We note that different choices of $\tau \in (0, 1]$ give different iterative updating rules for the algorithm. Once it is chosen in the initialization step, it is kept unchanged throughout the computation. Many values of $\tau$ have been tested. Listed in the table are results for $\tau = 1$, 0.95 and 0.01. It seems that the $\tau$ value at which the algorithm achieves its best performance is problem-dependent. But for this example, $\tau = 0.95$ is close to the best possible value of $\tau$ in terms of CPU time consumption in obtaining solutions of the same level of accuracy. The number $M$ in the artificial constraint should be as small as possible since larger $M$, in general, means more iterations in reaching an optimal solution.

We note that, for $n = 8$, [5] reported computational optimal solution $x^*$ of accuracies $1.0e - 3$ by a grid discretization method and $1.0e - 4$ by a cutting plane method. The new algorithm can obtain an optimal solution of accuracy $1.0e - 4$ with much less CPU time and much higher feasibility accuracy. We note that the numerical optimal solutions reported in [10] record a better accuracy for these cases, but the CPU consumption was ignored. Tests for larger $n$ have been done by different authors. For example, in [3], the cases $n = 10$, 20 and 30 were tested using a logarithm barrier cutting plane method. Even though the computational results show very satisfactory duality gaps for the subproblems, the feasibility of their solutions is unfortunately not guaranteed, as it solves a discretized subproblem. In comparison, the new algorithm is capable of finding solutions that have very accurate objective values with ignorable constraint violation for $1 \leq n \leq 16$. However, for $n \geq 17$, the problem seems to involve singularities and encounters more and more frequent ill-conditioned linear systems in computing the ladder points as $n$ increases.

Example 3. Find the best $L_1$ approximation to a given smooth function $b(t)$ from above by a Fourier sine series of order $\leq n$, in the form of $f_s(t) = \sum_{i=1}^{n} x_i \sin(it)$, over a given interval $[\alpha, \beta]$.

The problem is to determine the coefficient vector $x = [x_1; x_2, \cdots ; x_n]$ of $f_s(t)$ so that the $L_1$ distance between $f_s(t)$ and $b(t)$,

\[
\|f_s - b\|_1 = \int_{\alpha}^{\beta} |f_s(t) - b(t)| dt,
\]

is minimized subject to $f_s(t) \geq b(t)$ for all $t \in [\alpha, \beta]$.
In the following, we consider a special case where \( b(t) = -\sqrt{\pi t - t^2} \) and \([\alpha, \beta] = [0, \pi]\). Since, for \( f_s(t) \geq b(t) \),

\[
\|f_s - b\|_1 = \int_0^\pi |f_s(t) - b(t)| \, dt = c^n x + \frac{\pi^3}{8},
\]

where

\[
c = \left[\frac{1 - \cos(\pi)}{1}; \frac{1 - \cos(2\pi)}{2}; \cdots; \frac{1 - \cos(n\pi)}{n}\right].
\]

Thus, the problem is equivalent to the following LSIP problem.
Problem E₃.

\[
\begin{align*}
\min & \quad c^T x \\
\text{s. t.} & \quad a(t)x \leq d(t), \quad \text{for } t \in [0, \pi]
\end{align*}
\]

where \( c \) is as above,

\[
a(t) = -[\sin(t), \sin(2t), \ldots, \sin(nt)],
\]

and

\[
d(t) = \sqrt{\pi t - t^2}, \quad t \in [0, \pi].
\]

Table 2. Results for Example 3 with \( b(t) = -\sqrt{\pi t - t^2}, \quad 0 \leq t \leq \pi \) and parameters \( M = 5, \quad \varepsilon = 1.0e - 7, \quad \tau = 1 \)

| \( n \) | Number of iterations | CPU time | Maximum constraint violation \( \| f_s - b \|_1 \) | optimal \( L_1 \) distance \( \| f_s - b \|_1 \) |
|---|---|---|---|---|
| 5  | 21 | 1.078125 | 9.992007e - 15 | 0.200884 |
| 10 | 57 | 2.703125 | 5.206305e - 13 | 0.101011 |
| 15 | 65 | 2.687500 | 1.907363e - 13 | 0.052233 |
| 20 | 81 | 2.984375 | 9.769962e - 15 | 0.037951 |
| 25 | 98 | 5.328125 | 9.066787e - 14 | 0.025969 |
| 35 | 147 | 8.140625 | 7.838174e - 14 | 0.016149 |
| 55 | 210 | 11.859375 | 2.719638e - 12 | 0.008424 |
| 95 | 360 | 21.593750 | 1.731947e - 13 | 0.003782 |
| 200| 521 | 37.203125 | 2.856625e - 08 | 0.001253 |

Suppose that the optimal solution to problem E₃ is \( x^* \), then the required best \( L_1 \) approximation to \( b(t) \) over \([0, \pi]\) is achieved at \( f_s^*(t) = -a(t)x^* \) and the corresponding minimal distance is \( \| f_s - b \|_1 = c^T x^* + \pi^3/8 \).

Problem E₃ is solved for \( 1 \leq n \leq 200 \). Computational results for a number of \( n \) values are listed in Table 2. The plot of the computed optimal \( f_s(t) \) against that of \( b(t) \) is shown in Figure 2 with selected \( n \) values. The algorithm is very stable for these tested cases. For \( n \) close to 200, however, some effort has to be made to find an initial ladder that does not involve singularities (the linear system to be solved to obtain the initial ladder point is not ill-conditioned.)

Example 4. Find the best \( L_1 \) approximation to a given smooth function \( b(t) \) from above by a Fourier cosine series of order \( \leq n - 1 \), \( f_c(t) = \sum_{i=1}^{n} x_i \cos((i-1)t) \), over a given interval \([\alpha, \beta]\).

For this example, we take the same \( b(t) \) and the same interval \([\alpha, \beta] = [0, \pi]\) as in Example 3. Then the problem is equivalent to the following LSIP problem.

Problem E₄.

\[
\begin{align*}
\min & \quad c^T x \\
\text{s. t.} & \quad a(t)x \leq d(t), \quad \text{for } t \in [0, \pi]
\end{align*}
\]

where

\[
c = [\pi; 0; \cdots; 0],
\]

\[
a(t) = -[1, \cos(t), \cdots, \cos((n - 1)t)]
\]
and

\[ d(t) = \sqrt{\pi t - t^2} \text{ for } t \in [0, \pi]. \]

The problem is also solved, like in Example 3, for \(1 \leq n \leq 200\). Some computational results are listed in Table 3, and the corresponding calculated optimal functions \((f^*_c(t))\) are plotted in Figure 3. The algorithm is very efficient for all these cases given that an initial ladder that does not involve singularities is available for starting the algorithm. However, finding such an initial ladder has been proven to become increasingly more difficult when \(n\) increases. This difficulty is similar to what we encountered in solving Examples 2 and 3.
Table 3. Results for Example 4 with $b(t) = -\sqrt{\pi t - t^2}$, $0 \leq t \leq \pi$
and parameters $M = 5$, $\varepsilon = 1.0e - 7$, $\tau = 1$

| $n$ | Number of iterations | CPU time   | Maximum constraint violation $\| f_c - b \|_1$ |
|-----|---------------------|------------|-----------------------------------------------|
| 5   | 14                  | 0.718750   | 6.661338e - 15                                |
| 10  | 39                  | 1.765625   | 2.220446e - 16                                |
| 15  | 86                  | 3.140625   | 1.079370e - 12                                |
| 20  | 120                 | 5.031250   | 2.238653e - 12                                |
| 25  | 120                 | 6.421875   | 2.344791e - 13                                |
| 35  | 133                 | 7.281750   | 2.176808e - 10                                |
| 55  | 252                 | 14.125000  | 6.944051e - 14                                |
| 95  | 404                 | 23.906250  | 7.016610e - 14                                |
| 200 | 766                 | 55.468750  | 9.083422e - 08                                |

5. Summary. In this paper, we introduce the so-called generalized ladder point for linear semi-infinite programming and prove that a semi-infinite programming problem with a continuous linear inequality constraint system has an optimal solution if and only if it has one at a generalized ladder point. Based on this result, a centered climbing ladder algorithm is developed. The algorithm iteratively obtains a sequence of ladder points that converges to a feasible generalized ladder point – a special optimal solution. It combines a linear programming ladder algorithm with a very efficient one-dimensional global minimization method, namely the bridging method. The bridging method finds precisely the index point where the constraint is violated most at the current solution. It works faster than searching for the most violated constraint over a set of dense grid index points (used in discretization and cutting plane methods). Numerical tests show that the algorithm is very efficient.

Acknowledgments. The authors wish to thank Professor Miguel Angel Goberna for helping with the precise optimal solutions of Example 2 for the cases $1 \leq n \leq 8$.

REFERENCES

[1] E. J. Anderson and A. S. Lewis, An extension of the simplex algorithm for semi-infinite linear programming, Mathematical Programming, 44 (1989), 247–269.
[2] B. Betro, An accelerated central cutting plane algorithm for semi-infinite linear programming, Mathematical Programming, 101 (2004), 479–495.
[3] D. den Hertog, J. Kaliski, C. Roos and T. Terlaky, A logarithmic barrier cutting plane method for convex programming, Annals of Operations Research, 58 (1995), 69–98.
[4] M. C. Ferris and A. B. Philpott, An interior point algorithm for semi-infinite linear programming, Mathematical Programming, 43 (1989), 257–276.
[5] M. A. Goberna and M. A. López, Linear Semi-infinite Optimization, Wiley Series in Mathematical Methods in Practice, 2, John Wiley & Sons, Ltd., Chichester, 1998.
[6] M. A. Goberna, Linear semi-infinite optimization: Recent advances, in Continuous Optimization (eds. V. Jeyakumar and A. M. Rubinov), Appl. Optim., 99, Springer, New York, 2005, 3–22.
[7] R. Hettich, A review of numerical methods for semi-infinite optimization, in Semi-infinite Programming and Applications (eds. A. V. Fiacco and K. O. Kortanek), Lecture Notes in Econom. and Math. Systems, 215, Springer, Berlin, 1983, 158–178.
[8] R. Hettich, An implementation of a discretization method for semi-infinite programming, Mathematical Programming, 34 (1986), 354–361.
[9] R. Hettich and K. O. Kortanek, Semi-infinite programming: Theory, methods, and applications, SIAM Rev., 35 (1993), 380–429.
[10] S. Ito, Y. Liu and K. L. Teo, A dual parametrization method for convex semi-infinite programming, *Ann. Oper. Res.*, 98 (2000), 189–213.

[11] A. Kaplan and R. Tichatschke, Proximal interior point method for convex semi-infinite programming, *Optim. Methods Softw.*, 15 (2001), 87–119.

[12] Y. Liu, An exterior point method for linear programming based on inclusive normal cones, *J. Ind. Manag. Optim.*, 6 (2010), 825–846.

[13] Y. Liu, Duality theorem in linear programming: From trichotomy to quadrichotomy, *J. Ind. Manag. Optim.*, 7 (2011), 1003–1011.

[14] Y. Liu and K. L. Teo, A bridging method for global optimization, *J. Austral. Math. Soc. Ser. B*, 41 (1999), 41–57.

[15] Y. Liu, K. L. Teo and S. Y. Wu, A New quadratic semi-infinite programming algorithm based on dual parametrization, *J. Global Optim.*, 29 (2004), 401–413.

[16] R. Reemtsen, Discretization methods for the solution of semi-infinite programming problems, *J. Optim. Theory Appl.*, 71 (1991), 85–103.

[17] G. A. Watson, Lagrangian methods for semi-infinite programming problems, in *Infinite Programming*, Lecture Notes in Economics and Mathematical Systems (eds. E. J. Anderson and A. B. Philpott), Lecture Notes in Econom. and Math. Systems, 259, Springer, Berlin, 1985, 90–107.

[18] S. Y. Wu, S. C. Fang and C. J. Lin, Relaxed cutting plane method for solving linear semi-infinite programming problems, *J. Optim. Theory Appl.*, 99 (1998), 759–779.

Received November 2012; 1st revision February 2013; final revision July 2013.

E-mail address: yanqun.liu@rmit.edu.au
E-mail address: mingfang.ding@student.rmit.edu.au