Relaxed Cutting Plane Method for Solving Linear Semi-Infinite Programming Problems[1]

S.-Y. Wu[2], S. -C. Fang[3], and C. -J. Lin[4]

Communicated by Paul Tseng

[1] This research was partially supported by the North Carolina Supercomputing Center, Cray Research Grant, and the National Textile Center Research Grant.

[2] Professor, National Cheng Kung University, Department of Mathematics, Tainan, Taiwan, R.O.C.

[3] Walter Clark Professor, North Carolina State University, Industrial Engineering and Operations Research, Raleigh, North Carolina.

[4] Graduate Student, University of Michigan, Industrial and Operations Engineering, Ann Arbor, Michigan.

Abstract. One of the major computational tasks of using the traditional cutting plane approach to solve linear semi-infinite programming problems lies in finding a global optimizer of a non-linear and non-convex program. This paper generalizes Gustafson and Kortanek's scheme to relax this requirement. In each iteration, the proposed method chooses a point at which the infinite constraints are violated to a degree rather than at which the violation are maximized. A convergence proof of the proposed scheme is provided. Some computational results are included. An explicit algorithm which allows the unnecessary constraints to be dropped in each iteration is also introduced to reduce the size of computed programs.

Key Words: Linear semi-infinite programming, cutting plane method.

1 Introduction

Consider the following linear semi-infinite programming problem

$$(LSIP) \qquad \min \sum_{j=1}^{n} c_j x_j$$

$$\text{s.t.} \sum_{j=1}^{n} x_j f_j(t) \geq g(t), \forall t \in T \tag{1}$$

$$x_j \geq 0, \qquad j = 1, \cdots, n \tag{2}$$

where T is a compact metric space with an infinite cardinality, f_j, $j = 1, \cdots, n$, and g are real valued continuous functions defined on T. Note that T can be extended to a compact Hausdorff space (Refs. 1 and 2) without much difficulty.

Its dual problem can be formulated in the following form

$$(DLSIP) \qquad \max \int_T g(t) d\mu(t)$$

$$\text{s.t.} \int_T f_j(t) d\mu(t) \leq c_j, j = 1, \cdots, n \tag{3}$$

$$\mu \in M^+(T), \tag{4}$$

where $M^+(T)$ is the space of all non-negative bounded regular Borel measures on T.

Under some regularity conditions, it can be shown (Refs. 1 and 2) that there is no duality gap between $(LSIP)$ and $(DLSIP)$ and the latter achieves its optimum at an "extreme point" of its feasible domain. The applications of the linear semi-infinite programming can be referred to Refs. 3 and 4.

Many papers (Refs. 1, 3, 5 and 6) have dealt with solution methods for solving $(LSIP)$. According to a recent review article (Ref. 7), the so-called "cutting plane method," or "implicit exchange method," is one of the key solution techniques. Basically, this approach finds a sequence of optimal solutions of corresponding regular linear programs in a systematic way and shows that the sequence converges to an optimal solution of $(LSIP)$.

More precisely, in the kth iteration, let $T_k = \{t_1, t_2, \cdots, t_k\} \subset T$ and consider the following linear program

$$(LP_k) \qquad \min \sum_{j=1}^{n} c_j x_j$$

$$\text{s.t.} \sum_{j=1}^{n} x_j f_j(t_i) \ \geq \ g(t_i), i = 1, 2, \cdots, k \tag{5}$$

$$x_j \ \geq 0, \qquad\qquad j = 1, 2, \cdots, n. \tag{6}$$

Firstly, we solve (LP_k) for an optimal solution $\mathbf{x}^k = (x_1^k, x_2^k, \cdots, x_n^k)^T$. Then define

$$\phi_k(t) = \sum_{j=1}^{n} f_j(t) x_j^k - g(t). \tag{7}$$

Secondly, we find an optimizer

$$t_{k+1} \in \ arg \min_{t \in T} \phi_k(t). \tag{8}$$

If $\phi_k(t_{k+1}) = \sum_{j=1}^{n} f_j(t_{k+1}) x_j^k - g(t_{k+1}) \geq 0$, then $\mathbf{x}^k$ must be an optimal solution to $(LSIP)$ because (LP_k)'s feasible domain contains $(LSIP)$'s. Otherwise, we let $T_{k+1} = T_k \bigcup \{t_{k+1}\}$ to construct (LP_{k+1}) and continue the iterative process. The convergence proof of $\{\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^k, \cdots\}$ to an optimal solution of $(LSIP)$ can be found in Ref. 3. A refined version which allows us to drop some redundant points in T_k in order to reduce the size of (LP_k) can be found in Ref. 2.

In the above approach, one constraint (corresponding to one cut) is added at a time and the major computational work in each iteration involves (i) solving a linear program (LP_k) and (ii) finding a global minimizer t_{k+1} of $\phi_k(t)$. To reduce the computational requirement for solving (LP_k), an "inexact approach" was proposed earlier (Refs. 8 and 9). However, when the dimensionality of the compact metric space T becomes high, finding a minimizer of a continuous function $\phi_k(t)$ over T could be extremely time-consuming, in particular, when $f_j(t)$ and $g(t)$ are highly nonlinear and non-convex. In this case, the computational bottleneck of the cutting plane approach falls in finding a global minimizer t_{k+1} of $\phi_k(t)$.

Ideas of relaxing the requirement of finding the global minimizer for different settings can be found in Refs. 10 - 13. For these approaches, at each iteration, instead of finding the global minimizer t_{k+1} of $\phi_k(t)$, one either has to check that $\phi_k(t) \geq 0$, $\forall t \in T$, in order to add a constraint (Ref. 10), or has to find the value of $\delta(x^k) = min_{t \in T}\{\phi_k(t)\}$ in order to find a new cut at t_{k+1} such that $\phi_k(t_{k+1}) < 0$ and $\phi_k(t_{k+1}) \leq \delta(x^k) + \epsilon_k$, where $\{\epsilon_k\}$ is a given sequence of nonnegative numbers converging to zero as k increases to infinity (Refs. 12 and 13). Even so, the required computation could still be a bottleneck.

In this paper, we further relax Gustafson and Kortanek's scheme such that a new cut is found at any $t_{k+1} \in T$ with $\phi_k(t_{k+1}) < -\delta$, where $\delta > 0$ is a sufficiently small number which can be prescribed. In this way, we can avoid the task of finding the global minimizer t_{k+1} and/or checking the minimum value $\delta(x^k)$ in every iteration. After introducing the relaxed scheme in Section 2, we show that, under appropriate conditions, the proposed scheme terminates in finite iterations to generate an approximate solution with desired accuracy. Some computational experiments and analysis are included in Section 3. Based on the relaxed scheme, an explicit algorithm which allows the unnecessary constraints to be dropped in each iteration is developed in Section 4 while some concluding remarks are made in Section 5.

2 Relaxed Approach

Given that $\delta > 0$ is a prescribed small number, a general scheme is proposed as follows:

Step 1: Set $k \leftarrow 1$, choose any $t_1 \in T$, and set $T_1 = \{t_1\}$.

Step 2: Solve (LP_k) with an optimal solution $\mathbf{x}^k = (x_1^k, \cdots, x_n^k)^T$.

Define $\phi_k(t)$ according to (7).

Step 3: Find any $t_{k+1} \in T$ such that $\phi_k(t_{k+1}) < -\delta$.

If such t_{k+1} does not exist, stop and output $\mathbf{x}^k$ as the solution.

Otherwise, set $T_{k+1} = T_k \cup \{t_{k+1}\}$.

Step 4: Update $k \leftarrow k+1$ and go to Step 2.

Note that if (LP_k) is found to be infeasible in Step 2, then $(LSIP)$ is infeasible. In this case, there is no reason to continue the iterations. Therefore, without loss of generality, we may assume that (LP_k) is feasible in this paper. Also note that $t_{k+1} \notin T_k$ and if the scheme terminates in Step 3, then the output solution $\mathbf{x}^k$ indeed solves $(LSIP)$ with δ being small enough (up to machine accuracy, say 10^{-7}). More on the effects of the size of δ will be discussed later. Moreover, the linear dual of (LP_k) can be formulated as the following problem

$$(DLP_k) \qquad \max \sum_{i=1}^{k} g(t_i) y_i$$

$$\text{s.t.} \sum_{i=1}^{k} f_j(t_i) y_i \leq c_j, j = 1, 2, \cdots, n \tag{9}$$

$$y_i \geq 0, \qquad i = 1, 2, \cdots, k. \tag{10}$$

For the purpose of easy description of the proposed approach, we assume that (LP_k) is solvable with an optimal value denoted by $V(LP_k)$ and (DLP_k) is also solvable with an optimal value denoted by $V(DLP_k)$. Remember that $\mathbf{x}^k = (x_1^k, \cdots, x_n^k)^T$ solves (LP_k). Let $B_k = \{j_1^k, \cdots, j_{p_k}^k\}$ be an index set such that

$$x_j^k > 0, \quad \text{if and only if} \quad j \in B_k. \tag{11}$$

Then we have the following result:

Theorem 2.1: If (DLP_{k+1}) is nondegenerate, then $V(LP_{k+1}) > V(LP_k)$.

Proof: Define a $k \times n$ matrix A with $a_{ij} = f_j(t_i)$ being its (i,j)th element, for $i = 1, \cdots, k$, and $j = 1, \cdots, n$. Denoting $\mathbf{b} = (g(t_1), \cdots, g(t_k))^T$, we can rewrite (LP$_k$) in the following form:

$$
\begin{array}{lll}
(LP_k) & \min & \displaystyle\sum_{j=1}^{n} c_j x_j \\
& \text{s.t.} & \\
& & (A \mid -I_{k\times k}) \begin{pmatrix} \mathbf{x} \\ -- \\ \mathbf{x}' \end{pmatrix} = \mathbf{b}, \\
& & \mathbf{x}, \mathbf{x}' \geq 0.
\end{array}
$$

Let $\bar{\mathbf{x}}^k = \begin{pmatrix} \mathbf{x}^k \\ -- \\ \mathbf{x}'^k \end{pmatrix}$ be an optimal solution with a basis B. Its corresponding optimal basic variables are $\bar{\mathbf{x}}^k_B = B^{-1}\mathbf{b} \geq 0$. From the choice of t_{k+1} in Step 3 of the proposed scheme, we have

$$
\begin{aligned}
\phi_k(t_{k+1}) &= \textstyle\sum_{j=1}^{n} f_j(t_{k+1}) x_j^k - g(t_{k+1}) \\
&= \textstyle\sum_{j\in B_k} f_j(t_{k+1}) x_j^k - g(t_{k+1}) < -\delta.
\end{aligned} \tag{12}
$$

If we let $f = (f_1(t_{k+1}), \cdots, f_n(t_{k+1}))$, then (LP_{k+1}) becomes

$$
\begin{array}{lll}
(LP_{k+1}) & \min & \displaystyle\sum_{j=1}^{n} c_j x_j \\
& \text{s.t.} & \\
& & \begin{pmatrix} A & | & \\ -- & | & -I_{(k+1)\times(k+1)} \\ f & | & \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ -- \\ \mathbf{x}'' \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ -- \\ g(t_{k+1}) \end{pmatrix}, \\
& & \mathbf{x}, \mathbf{x}'' \geq 0.
\end{array}
$$

Denoting $\mathbf{c} = (c_1, \cdots, c_n)^T$, we have a linear dual program

$$(DLP_{k+1}) \qquad \max \qquad \sum_{i=1}^{k+1} g(t_i) y_i$$

$$\text{s.t.} \qquad \left(\begin{array}{c|c} A & \\ -- & -I_{(k+1)\times(k+1)} \\ f & \end{array} \right)^T \mathbf{y} \leq \left(\begin{array}{c} \mathbf{c} \\ -- \\ 0_{(k+1)\times 1} \end{array} \right).$$

Moreover, we let $f' = (f_1(t_{k+1}), \cdots, f_n(t_{k+1}), 0, \cdots, 0) \in R^{n+k}$ and $f_B' = (f_{j_1^k}(t_{k+1}), \cdots, f_{j_{p_k}^k}(t_{k+1}), 0, \cdots, 0)$ be those in f' corresponding to B. In this way, if

$$\left(\begin{array}{c|c} B & 0_{k\times 1} \\ -- & -- \\ f_B' & -1 \end{array} \right) \left(\begin{array}{c} \bar{\mathbf{x}}_B^k \\ -- \\ \mathbf{x}^* \end{array} \right) = \left(\begin{array}{c} \mathbf{b} \\ -- \\ g(t_{k+1}) \end{array} \right), \tag{13}$$

then, by (12),

$$\mathbf{x}^* = \sum_{j\in B_k} f_j(t_{k+1}) x_j^k - g(t_{k+1}) = \phi_k(t_{k+1}) < -\delta. \tag{14}$$

Moreover,

$$\left(\begin{array}{c} \bar{\mathbf{x}}_B^k \\ -- \\ \mathbf{x}^* \end{array} \right) = \left(\begin{array}{c|c} B & 0_{k\times 1} \\ -- & -- \\ f_B' & -1 \end{array} \right)^{-1} \left(\begin{array}{c} \mathbf{b} \\ -- \\ g(t_{k+1}) \end{array} \right). \tag{15}$$

Let $\mathbf{y}^* = (y_1^*, \cdots, y_k^*)^T = (B^{-1})^T c_B$, the theory of linear programming (Refs. 14 and 15) implies that $\mathbf{y}^*$ is an optimal solution of (DLP$_k$). Note that

$$\left(\left(\begin{array}{c|c} B & 0_{k\times 1} \\ -- & -- \\ f_B' & -1 \end{array} \right)^{-1} \right)^T \left(\begin{array}{c} c_B \\ -- \\ 0 \end{array} \right) = \left(\begin{array}{c} \mathbf{y}^* \\ -- \\ 0 \end{array} \right). \tag{16}$$

Hence we know that

$$\left(\begin{array}{c|c} A & \\ -- & -I_{(k+1)\times(k+1)} \\ f & \end{array} \right)^T \left(\begin{array}{c} \mathbf{y}^* \\ -- \\ 0 \end{array} \right) \leq \left(\begin{array}{c} \mathbf{c} \\ -- \\ 0_{(k+1)\times 1} \end{array} \right).$$

In other words, $\begin{pmatrix} \mathbf{y}^* \\ -- \\ 0 \end{pmatrix}$ is a feasible solution of (DLP_{k+1}) with a corresponding basic solution $\begin{pmatrix} \mathbf{x}_B^k \\ -- \\ \mathbf{x}^* \end{pmatrix}$ for (LP_{k+1}).

Now, since $\sum_{i=1}^k g(t_i)y_i^* + g(t_{k+1}) \cdot 0 = V(DLP_k)$, (DLP_{k+1}) is nondegenerate, and $\mathbf{x}^* < -\delta < 0$, the basic property of the dual simplex method (Ref. 15, p. 97) guarantees that

$$V(LP_{k+1}) = V(DLP_{k+1}) > V(DLP_k) = V(LP_k). \tag{17}$$

□

The incremental amount in Theorem 2.1 can be further analyzed. Let $\mathbf{y}^k = (y_1^k, \cdots, y_k^k)^T$ be an optimal solution of (DLP_k). We define a discrete measure μ_k on T such that

$$\mu_k(t) = \begin{cases} y_i^k, & \text{if } t = t_i \in T_k \\ 0, & \text{if } t \in T \setminus T_k. \end{cases} \tag{18}$$

In this way, $\mu_k(t) \geq 0, \ \forall t \in T$. Furthermore, let

$$T_k' = \{t \in T_k \mid \mu_k(t) > 0\} \tag{19}$$

and $B_k' = \{i_1^k, \cdots, i_{m_k}^k\}$ be an index set such that

$$\mu_k(t_i) > 0, \text{ if and only if } \ i \in B_k'. \tag{20}$$

In other words,

$$y_i^k > 0, \ \text{ if and only if } \ i \in B_k'. \tag{21}$$

We claim that

$$t_{k+1} \in T_{k+1}'. \tag{22}$$

Note that if $t_{k+1} \notin T_{k+1}'$, then $\mu_{k+1}(t_{k+1}) = 0$. In this case, the measure μ_{k+1} achieves nonzero value no more than at those points in T_k. Hence we have $V(DLP_{k+1}) = V(DLP_k)$ which contradicts Theorem 2.1. Therefore, without loss of generality, we can rearrange t_{k+1} to be the last element in T_{k+1}'. We define an $n \times m_k$ matrix H_k with its jth row vector being

$$(f_j(t_{i_1^k}), \cdots, f_j(t_{i_{m_k}^k})), \ \text{ for } j = 1, \cdots, n. \tag{23}$$

Since $\mu_k(t_i) > 0, \forall i \in B_k'$, from the "complementary slackness theorem" of linear programming (Refs. 14 and 15), we have

$$H_k^T \mathbf{x}^k = \mathbf{g}^k, \tag{24}$$

where

$$\mathbf{g}^k = (g(t_{i_1^k}), \cdots, g(t_{i_{m_k}^k}))^T. \tag{25}$$

Let M_k be a $p_k \times m_k$ matrix with its jth row vector being

$$(f_j(t_{i_1^k}), \cdots, f_j(t_{i_{m_k}^k})), \quad \text{for } j \in B_k. \tag{26}$$

Remember that t_{k+1} is the last element in T'_{k+1}. Now define an $n \times (m_k + 1)$ matrix H_{k+} with its jth row vector being

$$(f_j(t_{i_1^k}), \cdots, f_j(t_{i_{m_k}^k}), f_j(t_{k+1})), \quad \text{for } j = 1, \cdots, n. \tag{27}$$

For $\mathbf{g}^{k+1} = H_{k+}^T \mathbf{x}^{k+1}$, from the constraints satisfied, we know that

$$(\mathbf{g}^{k+1})^T \geq ((\mathbf{g}^k)^T, g(t_{k+1})) \equiv (\bar{\mathbf{g}}^{k+1})^T. \tag{28}$$

We define

$$\mathbf{s}^k = \mathbf{g}^{k+1} - \bar{\mathbf{g}}^{k+1}. \tag{29}$$

Then $\mathbf{s}^k \geq 0$. Moreover, the last element of $\mathbf{s}^k$ must be zero. Otherwise, we have $\mu_{k+1}(t_{k+1}) = 0$ which contradicts the fact that $\mu_{k+1}(t_{k+1}) > 0$. Remembering the definition of $\phi_k(t)$, on the dual side, we define

$$\phi_j^k = \sum_{i \in B'_k} f_j(t_i)\mu_k(t_i) - c_j, \text{ for } j = 1, \cdots, n. \tag{30}$$

Note that $\phi_j^k \leq 0$, $\forall j, k$, and we have the following result:

Theorem 2.2: If (DLP_{k+1}) is nondegenerate, then

$$V(LP_{k+1}) - V(LP_k) = \sum_{j=1}^{n} \phi_j^{k+1} x_j^k - \sum_{i \in B'_{k+1}} \phi_k(t_i)\mu_{k+1}(t_i)$$

$$= \sum_{j=1}^{m_k} s_j^k \mu_k(t_{i_j^k}) - \sum_{j=1}^{n} \phi_j^k x_j^{k+1}.$$

Proof: By the definition of $\phi_k(t)$, we have

$$\begin{aligned}
&\textstyle\sum_{i \in B'_{k+1}} \phi_k(t_i)\mu_{k+1}(t_i) \\
&= \textstyle\sum_{i \in B'_{k+1}} (\sum_{j=1}^{n} f_j(t_i)x_j^k - g(t_i))\mu_{k+1}(t_i) \\
&= \textstyle\sum_{i \in B'_{k+1}} \sum_{j=1}^{n} f_j(t_i)x_j^k \mu_{k+1}(t_i) - \sum_{i \in B'_{k+1}} g(t_i)\mu_{k+1}(t_i) \\
&= \textstyle\sum_{j=1}^{n} (\sum_{i \in B'_{k+1}} f_j(t_i)\mu_{k+1}(t_i))x_j^k - V(LP_{k+1}) \\
&= \textstyle\sum_{j=1}^{n} (\phi_j^{k+1} + c_j)x_j^k - V(LP_{k+1})
\end{aligned}$$

$$= \sum_{j=1}^{n} \phi_j^{k+1} x_j^k + \sum_{j=1}^{n} c_j x_j^k - V(LP_{k+1})$$
$$= \sum_{j=1}^{n} \phi_j^{k+1} x_j^k + V(LP_k) - V(LP_{k+1}).$$

Therefore,

$$V(LP_{k+1}) - V(LP_k) = \sum_{j=1}^{n} \phi_j^{k+1} x_j^k - \sum_{i\in B'_{k+1}} \phi_k(t_i)\mu_{k+1}(t_i).$$

On the other hand, by the definition of ϕ_j^k, we have

$$\sum_{j=1}^{n} \phi_j^k x_j^{k+1}$$
$$= \sum_{j=1}^{n} (\sum_{i\in B'_k} f_j(t_i)\mu_k(t_i) - c_j) x_j^{k+1}$$
$$= \sum_{j=1}^{n} (\sum_{i\in B'_k} f_j(t_i)\mu_k(t_i) x_j^{k+1}) - \sum_{j=1}^{n} c_j x_j^{k+1}$$
$$= \sum_{i\in B'_k} (\sum_{j=1}^{n} f_j(t_i) x_j^{k+1})\mu_k(t_i) - V(LP_{k+1})$$
$$= \sum_{j=1}^{m_k} (s_j^k + g(t_{i_j^k}))\mu_k(t_{i_j^k}) - V(LP_{k+1})$$
$$= \sum_{j=1}^{m_k} s_j^k \mu_k(t_{i_j^k}) + \sum_{i\in B'_k} g(t_i)\mu_k(t_i) - V(LP_{k+1})$$
$$= \sum_{j=1}^{m_k} s_j^k \mu_k(t_{i_j^k}) + V(LP_k) - V(LP_{k+1}).$$

Hence we have

$$V(LP_{k+1}) - V(LP_k) = \sum_{j=1}^{m_k} s_j^k \mu_k(t_{i_j^k}) - \sum_{j=1}^{n} \phi_j^k x_j^{k+1}.$$

□

Theorem 2.2 is a fundamental result, with it, we show that under proper conditions, for any given $\delta > 0$, the proposed scheme actually terminates in a *finite number of iterations.*

Theorem 2.3: Given any $\delta > 0$, in each iteration, assume that
(A1) (LP_k) has a bounded feasible domain,
(A2) (DLP_k) is nondegenerate.
Moreover, there exists a $\bar{\delta} > 0$ such that
(A3) $\mu_k(t_i) \geq \bar{\delta}, \quad \forall i \in B'_k$,
(A4) $\phi_j^k < -\bar{\delta}$, $\forall j \notin B_k$,
(A5) M_k^T has a square submatrix D_k with rank p_k $(= |B_k|)$ and $|\text{Det}(D_k)| > \bar{\delta}$.
Then, the proposed scheme terminates in a finite number of iterations.

Proof: Suppose that the scheme does not stop in a finite number of iterations. When (DLP_k) is nondegenerate for each k, Theorem 2.1 implies that

$$V(LP_1) < V(LP_2) < \cdots \leq V(LSIP).$$

Thus $\lim_{r\to\infty} V(LP_r) = \alpha \leq V(LSIP)$. We claim this is impossible.

By the assumption $(A1)$, the infinite sequence $\{\mathbf{x}^k\}$ is confined in a compact set C in R^n. There exists a subsequence $\{\mathbf{x}^{k_r}\}$ of $\{\mathbf{x}^k\}$ such that $\mathbf{x}^{k_r}$ converges to $\mathbf{x}^*$, and the subsequence $\{t_{k_r+1}\}$ converges to some point t_* as $r \to \infty$. Now we let

$$\phi_*(t) = \sum_{i=1}^{n} f_i(t)x_i^* - g(t). \tag{31}$$

Then $\phi_{k_r}(t_{k_r+1})$ converges to $\phi_*(t_*)$. Since $\phi_{k_r}(t_{k_r+1}) < -\delta$, for each r, we have $0 \neq \phi_*(t_*) \leq -\delta$.

Now let $\varepsilon \in (0, \delta)$ be an arbitrary number, we can find a large integer $N \in \{k_r\}_{r=1}^{\infty}$ such that

$$\mid V(LP_N) - \alpha \mid \leq \varepsilon^2 \text{ and } \mid \phi_N(t_{N+1}) - \phi_*(t_*) \mid \leq \varepsilon^2. \tag{32}$$

By Theorem 2.2, we have

$$\mid V(LP_{N+1}) - V(LP_N) \mid = \mid \sum_{j=1}^{m_N} s_j^N \mu_N(t_{i_j^N}) - \sum_{j=1}^{n} \phi_j^N x_j^{N+1} \mid \leq \varepsilon^2 \tag{33}$$

Remember that $\phi_j^N \leq 0$, for $j = 1, 2, \cdots, n$, each term in the first summation sign of (33) is nonnegative, and each term in the second summation sign of (33) is nonpositive. It follows that $0 \leq s_j^N \mu_N(t_{i_j^N}) \leq \varepsilon^2$, for $j = 1, \cdots, m_N$, and $0 \leq -\phi_j^N x_j^{N+1} \leq \varepsilon^2$, for $j = 1, \cdots, n$. By the assumption $(A3)$, we have $\mu_N(t_i) \geq \bar{\delta}, \forall i \in B_N'$. Hence

$$s_j^N \leq \varepsilon, \qquad \text{for } j = 1, \cdots, m_N. \tag{34}$$

By the assumption $(A4)$, we have

$$\phi_j^N < -\bar{\delta}, \qquad \text{if } j \notin B_N. \tag{35}$$

It follows from (33) that

$$x_j^{N+1} \leq \varepsilon, \qquad \text{if } j \notin B_N. \tag{36}$$

By (29), (34), and (36), we have

$$\sum_{j \in B_N} f_j(t_i) x_j^{N+1} = g(t_i) + O_i(\varepsilon), \qquad \text{for } i \in B'_N, \tag{37}$$

where $O_i(\varepsilon) \to 0$ as $\varepsilon \to 0$.

By (24), we have

$$\sum_{j \in B_N} f_j(t_i) x_j^N = g(t_i), \qquad \text{for } i \in B'_N. \tag{38}$$

By the definition of M_N, we know that the matrix M_N has row vectors

$$(f_j(t_{i_1^N}), f_j(t_{i_2^N}), \cdots, f_j(t_{i_{m_N}^N})), \qquad \text{for } j \in B_N.$$

Let $\underline{\mathbf{x}} = (x_j^N)_{j \in B_N}$ and $\underline{\mathbf{x}}' = (x_j^{N+1})_{j \in B_N}$, then (37) and (38) can be expressed as

$$M_N^T \underline{\mathbf{x}}' = (g(t_{i_1^N}) + O_1(\varepsilon), \cdots, g(t_{i_{m_N}^N}) + O_{m_N}(\varepsilon))^T$$

and

$$M_N^T \underline{\mathbf{x}} = (g(t_{i_1^N}), \cdots, g(t_{i_{m_N}^N}))^T.$$

It follows from the assumption $(A5)$ that

$$| \, x_j^{N+1} - x_j^N \, | < \varepsilon^*(\varepsilon), \quad \text{for } j \in B_N, \tag{39}$$

where $\varepsilon^*(\varepsilon) \to 0$ as $\varepsilon \to 0$.

Combining the facts that $\phi_{N+1}(t_{N+1}) = 0$, (36), and (39), we have

$$\phi_N(t_{N+1}) \to 0 \text{ as } \varepsilon \to 0. \tag{40}$$

But $\phi_N(t_{N+1}) \to \phi_*(t_*) \neq 0$, as $N \to \infty$. Hence (40) cannot be true and we have a contradiction. Therefore our claim is valid and the proof is complete.

$\square$

Note that (A1) is commonly assumed in linear semi-infinite programming to simplify proofs. It can be relaxed by using "bounded level sets." (A2) is also a technical condition commonly used in linear programming. Moreover, when δ is chosen to be sufficiently small, (A3), (A4), and (A5) in general can be satisfied without much difficulty. The violation of any of these three assumptions will lead to some rare instances of degeneracy or inconsistency. In particular, the violation of (A3) will result in a subsequence of (DLP_k) which has a limit optimal solution being dual degenerate. Similar situation goes for the violation of (A4). Moreover, the violation of (A5) will provide a subsequence of (LP_k) whose determinant value of optimal basis matrix tends to zero.

Under these conditions, Theorem 2.3 assures that the proposed scheme terminates in finitely many iterations, say k^* iterations, with an optimal solution $\mathbf{x}^{k^*} = (x_1^{k^*}, \cdots, x_n^{k^*})^T$, such that $x_j^{k^*} > 0$, for $j \in B_{k^*}$. In this case, $\mathbf{x}^{k^*}$ is of course feasible for $(LSIP)$, and it can be viewed as an approximate solution of $(LSIP)$. The next theorem tells us how good such approximate solution can be.

Theorem 2.4: For any given $\delta > 0$, if there exists $\bar{\mathbf{x}} = (\bar{x}_1, \cdots, \bar{x}_n)^T$ with $\bar{x}_j \geq -x_j^{k^*}/\delta,\ \forall j \in B_{k^*}$, and $\bar{x}_j \geq 0,\ \ \forall j \notin B_{k^*}$, such that

$$\sum_{j=1}^{k} \bar{x}_j f_j(t) \geq 1, \quad \forall t \in T, \tag{41}$$

then

$$|V(LP_{k^*}) - V(LSIP)| \leq \delta |\sum_{j=1}^{n} c_j \bar{x}_j|.$$

Proof: By the definition of $\mathbf{x}^{k^*}$, we have

$$\sum_{j=1}^{n} f_j(t) x_j^{k^*} - g(t) \geq -\delta, \qquad \text{for each } t \in T. \tag{42}$$

By (41), we have

$$\sum_{j=1}^{n} f_j(t) \delta \bar{x}_j \geq \delta, \qquad \text{for each } t \in T. \tag{43}$$

It follows from (42) and (43) that

$$\sum_{j=1}^{n} f_j(t)(x_j^{k^*} + \delta \bar{x}_j) - g(t) \geq 0, \qquad \text{for each } t \in T. \tag{44}$$

By our assumption, we know that $x_j^{k^*} + \delta \bar{x}_j \geq 0$, for $j = 1, 2, \cdots, n$. Hence $\mathbf{x}^{k^*} + \delta \bar{\mathbf{x}}$ is a feasible solution of $(LSIP)$. Therefore,

$$| V(LP_{k*}) - V(LSIP) | \quad \leq \quad | \sum_{j=1}^{n} c_j x_j^{k^*} - \sum_{j=1}^{n} c_j (x_j^{k^*} + \delta \bar{x}_j) | = \delta | \sum_{j=1}^{n} c_j \bar{x}_j | .$$

$\square$

Note that, the required condition (41) implies that (1) has an interior solution. Also when $\delta > 0$ is chosen to be small enough, since $\{f_j(t) \mid j \in B_{k*}\}$ are in general linearly independent, the existence of the required $\bar{\mathbf{x}}$ in Theorem 2.4 is not a problem. Theorems 2.3 and 2.4 guarantee that the proposed scheme generates an approximate solution to $(LSIP)$ with any level of accuracy in a finite number of iterations.

3 Computational Experiments

In this section, the following two commonly seen L1 problems (Refs. 3 and 16) are used to illustrate the computational behavior of the proposed scheme:

Problem 3.1

$$\begin{aligned} &\max \sum_{j=1}^{8} \frac{-w_j}{j} \\ &\text{s.t.} \quad \sum_{j=1}^{8} -t^{j-1} w_j \leq \frac{-1}{2-t}, \quad t \in [0,1]. \end{aligned}$$

Problem 3.2

$$\begin{aligned} &\max \sum_{j=1}^{7} \frac{-w_j}{j} \\ &\text{s.t.} \quad \sum_{j=1}^{7} -t^{j-1} w_j \leq \sum_{j=0}^{4} t^{2j}, \quad t \in [0,1]. \end{aligned}$$

It was reported in Refs. 3 and 16 that 0.6931 and -1.78688 are approximate optimal solutions to Problem 1 and Problem 2, respectively.

We use MATLAB version 4.2c (Ref. 17) on a SUN UltraSparc1 workstation for the experiment. Three different approaches, namely, the traditional cutting plane method, the proposed method, and the well-known discretization method, have been implemented.

For all three methods, the **lp** subroutine of the MATLAB optimization toolbox was used to solve linear programs in each of them. For the discretization method, we discretized the interval $[0, 1]$ into 101 evenly spaced points and solved a linear program with 101 explicit constraints. For the traditional cutting plane method, the **fmin** subroutine of MATLAB was utilized for finding the global minimizer of problem (8).

For the proposed method, we set $\delta = 0.0001$ and recursively discretize T to find t_{k+1} in Step 3. In other words, in the beginning, the interval $[0, 1]$ was discretized by 11 evenly spaced points and test each point to see if $\phi(t) < -\delta$. If all points failed, then we refine the discretization by 101 points to test. If this failed, then we test 1001 points. If this failed again, the MATLAB subroutine **fmin** is finally employed to find t_{k+1}. Our numerical experiment has shown that only in the last iteration, the subroutine **fmin** is called for the proposed method.

In our experiment, the following condition

$$\phi_k(t_{k+1}) > -0.0001$$

was used as a stopping criterion for both the traditional and proposed methods. Also notice that the relaxed scheme was stated in Section 2 as a theoretical algorithm, without loss of generality, under the assumption that (LP_k) is solvable with an optimal solution $\mathbf{x}^k$. In practice, even when $(LSIP)$ is solvable, some (LP_k) at the early stages may still be unbounded below. This problem can be easily handled by adding enough points to T_1 to start the first iteration. Or, we can simply choose $\mathbf{x}^k$ in Step 2 to be a feasible solution of (LP_k), whose objective value is negatively large. In our experiments, we adopted the latter approach. The numerical results are shown in the following table:

Table 1: Comparison of different methods

	Problem 3.1			Problem 3.2		
	obj. value	k	*time* (sec)	*obj. value*	k	*time* (sec)
Traditional cp method	0.693146	14	6.87	-2.2395e+11*	71	40.96
Proposed cp method	0.693148	12	4.82	-1.786917	14	4.94
Discretized method	0.693148		6.18	-1.786901		1.46

* : the algorithm does not converge in 70 iterations

In the table, *obj. value* is the final objective value, k indicates the number of iterations required, and *time* is in cpu-seconds. Note that k also represents the number of linear programs (or number of explicit constraints) solved. For both problems, the proposed method generates a better approximate solution (comparable to the result obtained by the discretization method with 100 constraints) in shorter time than the traditional cutting plane method. For the second problem, the traditional cutting plane method failed to find an approximate solution in 70 iterations. In this case all optimizers, i.e., $t_k's$, generated by the traditional cutting plane method stay quite close, so the traditional method failed to converge fast. This expriment shows the potential of the proposed method.

To further understand the role of δ played in the proposed method, we ran the proposed method for both problems with different δ value. The results are shown in the following table:

Table 2: Numerical behaviors of using different δ

	Problem 3.1			Problem 3.2		
δ	*obj. value*	k	*time* (sec)	*obj. value*	k	*time* (sec)
0.0001	0.693148	12	4.82	-1.786917	14	4.94
0.01	0.693148	12	4.75	-1.787018	10	2.95
1.0	0.693148	11	4.46	-1.787018	9	2.75

Although in theory for the proposed method, we think δ should be very small, such as 10^{-7} for our workstation implementation. But in practice, δ can be chosen much larger than we expected. In our experiment, the results are not bad even when $\delta = 1.0$.

4 Explicit Algorithm

The proposed scheme adds one inequality constraint in each iteration. Hence the size of (LP_k) becomes larger and larger. In order to avoid solving too large problems, we would like to exploit the possibility of dropping unnecessary constraints while a new constraint is added in each iteration. This is referred to as an "explicit exchange method" in this paper.

To introduce such an explicit algorithm, let us start with some notation. Let $T' = \{t_1, ..., t_m\}$ be a subset with m elements in T. We denote the following linear program with m explicit constraints induced by T' as

$$LP(T') \qquad \min \quad \sum_{j=1}^{n} c_j x_j$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_j f_j(t_i) \ \geq \ g(t_i), i = 1, 2, \cdots, m \tag{45}$$

$$x_j \ \geq 0, \qquad j = 1, 2, \cdots, n. \tag{46}$$

Similar to situation faced in Section 2, for the purpose of easy description, given that $\delta > 0$ is a prescribed small number, we state our explicit algorithm in the following steps, under the assumption that $LP(T_k)$ and its dual $DLP(T_k)$ are both solvable:

Step 0: Let $k \leftarrow 1$, choose any $t_1^0 \in T$, set $T_1 = \{t_1^0\}$, and set $m_0 = 0$.

Step 1: Solve $LP(T_k)$. Let $\mathbf{x}^k = (x_1^k, \ldots, x_n^k)^T$ be an optimal solution.

Define $\phi_k(x)$ according to Equation (7).

Step 2: Solve $DLP(T_k)$. Let $\mathbf{y}^k = (y_1^k, \ldots, y_{m_{k-1}+1}^k)^T$ be an optimal solution.

Define a discrete measure μ_k on T by letting

$$\mu_k(t) = \begin{cases} y_i^k(\geq 0) & \text{if } t = t_i^{k-1} \in T_k, \\ 0 & \text{if } t \in T \backslash T_k. \end{cases}$$

Set $E_k = \{t \in T_k \mid \mu_k(t) > 0\} = \{t_1^k, \cdots, t_{m_k}^k\}$.

Step 3: Find any $t_{m_k+1}^k \in T$ such that $\phi_k(t_{m_k+1}^k) < -\delta$.

If such $t_{m_k+1}^k$ does not exist, stop and output $\mathbf{x}^k$ as a solution.

Otherwise, set $T_{k+1} = E_k \cup \{t_{m_k+1}^k\}$.

Step 4: Update $k \leftarrow k + 1$ and go to Step 1.

Note that $t_{m_k+1}^k \notin T_k$ and if the algorithm terminates in Step 3, then the output solution $\mathbf{x}^k$ indeed solves $(LSIP)$ with δ being sufficiently small. Also note that when it is applicable, a primal-dual algorithm can be employed to find solutions for $LP(T_k)$ and $DLP(T_k)$ simultaneously for computational efficiency.

Remember that $\mathbf{x}^k = (x_1^k, \cdots, x_n^k)^T$ solves $LP(T_k)$. Let $B_k = \{j_1^k, \cdots, j_{p_k}^k\}$ be an index set such that $x_j^k > 0$, if and only if $j \in B_k$. Also let M_k be a $p_k \times m_k$ matrix with its jth row vector being

$$(f_j(t_1^k), \cdots, f_j(t_{m_k}^k)), \quad \text{for } j \in B_k. \tag{47}$$

On the dual side, remembering the definition of $\phi_k(t)$, we define

$$\phi_j^k = \sum_{i=1}^{m_k} f_j(t_i^k)\mu_k(t_i^k) - c_j, \text{ for } j = 1, \cdots, n. \tag{48}$$

In this way, we know that $\phi_j^k \leq 0$, $\forall j, k$.

With this setting and parallel to the proof given in Theorem 2.3, we have the following result:

Theorem 4.1: Given any $\delta > 0$, in each iteration, assume that
(A1) The set $\{\mathbf{x}^1, \mathbf{x}^2, \ldots\}$ is bounded.

(A2) $DLP(T_k)$ is nondegenerate.
Moreover, there exists a $\bar{\delta} > 0$ such that
(A3) $\mu_k(t_i^k) \geq \bar{\delta}, \quad \forall i = 1, ..., m_k,$
(A4) $\phi_j^k < -\bar{\delta}, \forall j \notin B_k,$
(A5) M_k^T has a square submatrix D_k with rank p_k $(= |B_k|)$ and $|\text{Det}(D_k)| > \bar{\delta}$.
Then, the explicit algorithm terminates in a finite number of iterations.

The above theorem asserts that the explicit algorithm terminates with a solution in a finite number of iterations, under proper assumptions. Similarly, the result of Theorem 2.4 can be used to tell how good such a solution is.

Note that, for the relaxed scheme in Theorem 2.3, if (LP_k) has a bounded feasible domain, so do (LP_{k+1}), $(LP_{k+2}), \ldots$. However, for the explicit algorithm, T_k may not be a subset of T_{k+1}. Therefore, we directly assume that the set of optimal solutions to $LP(T_k)$, $k = 1, 2, \ldots$, is bounded.

We now use Problem 2 of Section 3 to illustrate the potential of the explicit algorithm. The computational experiment was conducted in the same test environment as described in Section 3, and the result is shown in the following table:

Table 3: Comparison of relaxed scheme and the explicit algorithm

	Number of constraints in $LP(T_k)$	
iteration k	relaxed scheme	explicit algorithm
1-9	1-9	1-9
10	10	6
11	11	6
12	12	6
13	13	6
14	14	6
$time$(sec)	4.94	4.73
$obj.\ value$	-1.786917	-1.786917

Observe that in the first nine iterations, since $LP(T_k)$ is unbounded below and hence $DLP(T_k)$ is infeasible, we simply find one feasible solution of $LP(T_k)$ and define $\phi_k(x)$ in Step 1, set $E_k = T_k$, and jump to Step 3. In this way, both methods add one constraint in iteration. After the $9th$ iteration, $DLP(T_k)$ becomes feasible and the explicit algorithm starts dropping unnecessary constraints. In this experiment, both cases stop at the $14th$ iteration with literally the same final objective value. Although the time reduction is not huge for this small problem, the potential of the explicit algorithm is clearly seen.

5 Concluding Remarks

In this paper, we have presented a relaxed cutting plane scheme to solve linear semi-infinite programming problems. The proposed scheme, in each iteration, chooses a point where the infinite constraints are violated to a degree rather than a point where the violation are maximized to generate a new cut for the next iteration. Under proper conditions, it has been proven that the proposed scheme can generate an approximate solution of any level of accuracy in a finite number of iterations. Based on this scheme, an explicit algorithm which allows the dropping of unnecessary constraints is developed.

Since the requirement of finding an optimizer of a nonlinear and nonconvex program is relaxed, we see the potential advantage of the proposed scheme, in particular, in higher dimensional spaces. Our very limited computational results support the theory. Although the method we implemented for finding t_{k+1} in Step 3 of the proposed scheme is very primitive, and may not be good for general purposes, the potential advantage of the proposed method is clearly illustrated.

References

1. ANDERSON, E. J., and NASH, P., *Linear Programming in Infinite-Dimensional Spaces,* Wiley, Chichester, England, 1987.
2. LAI, H.-C., and WU, S.-Y., *On Linear Semi-Infinitec Programming Problems: an Algorithm,* Numerical Functional Analysis and Optimization, Vol. 13, pp. 287-304, 1992.
3. GLASHOFF, K., and GUSTAFSON, S. Å., *Linear Optimization and Approximation,* Springer Verlag, New York, New York, 1982.
4. GUSTAFSON, S. Å., and KORTANEK, K. O., *Semi-Infinite Programming and Applications,* Mathematical programming: The State and Art. Edited by A. Bachem, M. Grotschel and B. Korte, Springer Verlag, Berlin, Germany, pp. 132-157, 1983.
5. GUSTAFSON, S. Å., *On The Computational Solution of a Class of Generalized Moment Problems,* SIAM Journal on Numerical Analysis, Vol. 7, pp. 343-357, 1970.
6. GUSTAFSON, S. Å., and KORTANEK, K. O., *Numerical Treatment of a Class of Semi-Infinite Programming Problems,* Naval Research Logistics Quarterly, Vol. 20, pp. 473-504, 1973.
7. HETTICH, R., and KORTANEK, K. O., *Semi-Infinite Programming: Theory, Method and Applications,* SIAM Review, Vol. 35, pp. 380-429, 1993.
8. FANG, S.-C., and WU, S.-Y., *An Inexactc Approach to Solving Linear Semi-Infinite Programming Problems,* Optimization, Vol. 28, pp. 291-299, 1994.
9. FANG, S.-C., LIN, C.-J., and WU, S.-Y., *On Solving Convex Quadratic Semi-Infinite Programming Problems,* Optimization, Vol. 31, pp. 107-125, 1994.
10. GRIBIK, P. R., *A Central Cutting Plane Algorithm for SIP,* Semi-infinite Programming. Edited by R. Hettich, Springer Verlag, Berlin, Germany, pp. 66-82, 1979.
11. TICHATSCHKE, R., and NEBELING, V., *A Cutting Plane Algorithm for Solving Quadratic Semi-Infinite Programs,* Optimization, Vol. 19, pp. 803-817, 1988.
12. GUSTAFSON, S. Å., and KORTANEK, K. O., *Computation of Optimal Experimental Designs for Air Quality Surveillance,* Quantitative Modelle für Ökonomisch-Ökologische Analysen. Edited by P. J. Jansen, O. Moeschlin, and O. Rentz, Verlag Anton Hain, Meisenheim, Germany, pp. 43-60, 1976.
13. GUSTAFSON, S. Å., and KORTANEK, K. O., *Computational Schemes for Semi-Infinite Programs,* Technical Report, Department of Mathematics, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1977.
14. FANG, S.-C., and PUTHENPURA, S. C., *Linear Optimization and Extensions: Theory and Algorithms,* Prentice Hall, Englewood Cliffs, New Jersey, 1993.
15. LUENBERGER, D. G., *Linear and Nonlinear Programming,* Addison-Wesley, Reading, Massachusetts, 1984.
16. FERRIS, M. C., and PHILPOTT, A. B., *An Interior Point Algorithm for Semi-Infinite Linear Programming,* Mathematical Programming, Vol. 43, PP. 257-276, 1989.
17. MATLAB User Guide, The Math Works, Natick, Massachusetts, 1994.