ELSEVIER

Journal of Computational and Applied Mathematics 61 (1995) 87–103

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

Implementation of an inexact approach to solving linear semi-infinite programming problems

Chih-Jen Lin[a], Eugene K. Yang[b,1], Shu-Cherng Fang[c,*,2], Soon-Yi Wu[d]

[a] *Department of Mathematics, National Taiwan University, Taipei, Taiwan, ROC*
[b] *Institute of Applied Mathematics, National Tsing-Hua University, Hsinchu, Taiwan, ROC*
[c] *Operations Research and Industrial Engineering, North Carolina State University, Raleigh, NC 27695-7913, United States*
[d] *Institute of Applied Mathematics, National Cheng-Kung University, Tainan, Taiwan, ROC*

Received 9 November 1993; revised 18 April 1994

Abstract

In this paper, we implement an extended version of the inexact approach proposed by Fang and Wu (1994) for solving linear semi-infinite programming problems. Some interesting numerical results are reported. The results confirm that the inexact approach is indeed more efficient and more robust than the exact approach.

Keywords: Semi-infinite programming; Linear programming; Entropy optimization

1. Introduction

Consider the following linear semi-infinite programming problems (LSIP):

$$\text{Minimize} \quad \sum_{j=1}^{n} c_j x_j$$

$$\text{subject to} \quad \sum_{j=1}^{n} x_j f_j(t) \geqslant g(t) \quad \forall t \in T, \tag{1.1}$$

$$x_j \geqslant 0, \quad j = 1, 2, \ldots, n, \tag{1.2}$$

* Corresponding author. E-mail: fang@eos.ncsu.edu.

[1] Partially supported by the National Science Council Grants #NCS82-0208-M-007-055 and #NCS83-0208-M-007-119 of the Republic of China.

[2] Partially supported by the NCSC-Cray Research Grant and the National Science Council Grant #NCS81-0415-E-007-10 of the Republic of China.

SSDI 0377-0427(94)00051-2

where T is a compact metric space with an infinite cardinality, f_j $(j = 1, 2, \ldots, n)$ and g are real valued continuous functions defined on T. Here LSIP means a linear optimization problem with a finite number of variables and infinitely many constraints. When T has a finite cardinality, the problem is reduced to a regular linear programming problem. On the other hand, T can be extended to a compact Hausdorff space [1, 16] or a measure space [17] without much difficulty.

A dual program of LSIP can be formulated as the following problem (DLSIP):

$$\text{Maximize} \quad \int_T g(t)\,\mathrm{d}v(t)$$

$$\text{subject to} \quad \int_T f_j(t)\,\mathrm{d}v(t) \leqslant c_j, \quad j = 1, \ldots, n, \tag{1.3}$$

$$v \in M^+(T), \tag{1.4}$$

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

When LSIP has a nonempty feasible region under some regularity conditions, it has been shown [1, 16] that LSIP achieves its optimum at least at an "extreme point" of its feasible region. In this case, there is no duality gap between LSIP and DLSIP. For applications of LSIP refer to [12, 16].

There are many papers [1, 8–13, 16] dealing with solution methods for solving LSIP. One framework due to Gustafson and Kortanek [13] finds a sequence of optimal extreme points of corresponding regular linear programming problems in a systematic way and shows that the sequence converges to an optimal solution of LSIP. To be more precise, at the kth iteration, let $T_k = \{t_1, t_2, \ldots, t_k\}$ be a subset of T with k points in it and consider the following linear program subproblem (LP_k):

$$\text{Minimize} \quad \sum_{j=1}^{n} c_j x_j$$

$$\text{subject to} \quad \sum_{j=1}^{n} f_j(t_i) x_j \geqslant g(t_i), \quad i = 1, 2, \ldots, k, \tag{1.5}$$

$$x_j \geqslant 0, \quad j = 1, 2, \ldots, n. \tag{1.6}$$

Then solve LP_k to find an optimal extreme point $\boldsymbol{x}^k = (x_1^k, x_2^k, \ldots, x_n^k)^{\mathrm{T}}$. Let

$$t_{k+1} = \underset{t \in T}{\text{minimizer}} \left\{ \sum_{j=1}^{n} f_j(t) x_j^k - g(t) \right\}. \tag{1.7}$$

If $\sum_{j=1}^{n} f_j(t_{k+1}) x_j^k - g(t_{k+1}) \geqslant 0$, then $\boldsymbol{x}^k$ must be an optimal solution to LSIP. Otherwise, let $T_{k+1} = T_k \cup \{t_{k+1}\}$ to continue this iterative process. It was shown in [12] that the sequence $\{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^k, \ldots\}$ converges to an optimal solution of LSIP.

With recent advances in the interior-point approach to solving linear programming problems [15, 4, 6, 7], Fang and Wu [9] proposed replacing LP_k in the aforementioned framework by an

entropic perturbation problem LP_{μ_k}:

$$\text{Minimize} \quad \sum_{j=1}^{n} c_j x_j + \mu_k \sum_{j=1}^{n} x_j \ln x_j$$

$$\text{subject to} \quad \sum_{j=1}^{n} f_j(t_i)\, x_j \geqslant g(t_i), \quad i = 1, 2, \ldots, k, \tag{1.8}$$

$$x_j \geqslant 0, \quad j = 1, 2, \ldots, n, \tag{1.9}$$

where $\mu_k > 0$ is a perturbation parameter. Under appropriate assumptions, it was shown in [9] that the sequence of optimal solutions $\{x(\mu_k) | x(\mu_k)$ solves LP_{μ_k}, $k = 1, 2, \ldots\}$ converges to an optimal solution of LSIP as μ_k approaches zero.

Note that LP_{μ_k} and LP_k share the same feasible region. When the regions of LP_k are bounded sets in $\mathbb{R}^n$ for k greater than or equal to a positive integer K (i.e., the Bounded Feasible Region Assumption), it was shown in [7] that LP_{μ_k} becomes LP_k as μ_k approaches zero. Hence, LP_{μ_k} can be viewed as a "relaxation" of LP_k, and $x(\mu_k)$ is an "inexact" optimal solution to LP_k. Therefore, it is called an "inexact approach". Note that we slightly generalize the Bounded Feasible Region Assumption of [9], in which $K = 1$, to include more problems arising in practice.

In this paper, we focus on the implementation issues and develop an extended version of Fang and Wu's inexact approach [9] for linear semi-infinite programming. In Section 2, we state the inexact algorithm proposed in [9] and extend it with another layer of inexactness. The key implementation issues are discussed in Section 3 and numerical results are reported in Section 4. Some concluding remarks are made in Section 5.

2. Algorithms

Let us denote the feasible region of LSIP by F and its interior (relative to $x \geqslant 0$) by F^0. In this paper, we assume that $F^0 \neq \emptyset$ (i.e., the Interior Point Assumption) and the Bounded Feasible Region Assumption holds. A version of the exact approach which is virtually the same as the one as proposed in [9] can be described as follows:

Algorithm 1.

Step 1: Set $k = 1$, choose $t_1 \in T$, set $T_1 = \{t_1\}$, choose $\eta_1, \eta_2, \eta_3, \eta_4 > 0$ (depending on machine accuracy) to be sufficiently small numbers, and

$$0 < \delta_1 < 1, \quad \mu_1 > 0.$$

Step 2: Find the optimal solution $x(\mu_k)$ of LP_{μ_k}, by using η_2 as the stopping tolerance in [7]. Let

$$\phi_{k+1}(t) = \sum_{j=1}^{n} f_j(t) x_j(\mu_k) - g(t) \quad \forall t \in T. \tag{2.1}$$

Find a minimizer t_{k+1} of $\phi_{k+1}(t)$ over T and calculate

$$\phi_{k+1}(t_{k+1}) = \sum_{j=1}^{n} f_j(t_{k+1})\, x_j(\mu_k) - g(t_{k+1}).$$

Step 3: If $\phi_{k+1}(t_{k+1}) \geqslant -\eta_3$, then check μ_k.
If $\mu_k \leqslant \eta_1$, then Stop! $\boldsymbol{x}(\mu_k)$ is optimal to LSIP.
Otherwise, Go to Step 4.
If $\phi_{k+1}(t_{k+1}) < -\eta_3$, Go to Step 4. ($\boldsymbol{x}(\mu_k)$ is not feasible to LSIP.)
Step 4: Let $T_{k+1} = T_k \cup \{t_{k+1}\}$.
Update μ_k: if $\mu_k > \eta_4$, then set $\mu_{k+1} = \delta_1 \mu_k$; otherwise, set $\mu_{k+1} = \mu_k$.
Reset $k \leftarrow k + 1$ and Go to Step 2.

Note that in the early stages of Algorithm 1 (i.e., when k is small), for a given μ_k, it is not absolutely necessary to find a "precise solution" for LP_{μ_k}. An "approximate solution" may do the job. Therefore, we extend Algorithm 1 by adding another layer of inexactness, namely, LP_{μ_k} is considered to be solved when its optimality condition is satisfied within a tolerance ε_k. By reducing ε_k to zero, as k goes to infinity, we return to the original design of Algorithm 1. This extension is outlined as follows:

Algorithm 2.
Step 1: Set $k = 1$, choose $t_1 \in T$, set $T_1 = \{t_1\}$, choose $\eta_1, \eta_2, \eta_3, \eta_4 > 0$ (depending on machine accuracy) to be sufficiently small numbers, and

$$0 < \delta_1, \delta_2 < 1, \quad \mu_1, \varepsilon_1 > 0.$$

Step 2: Find an approximate optimal solution to LP_{μ_k} and denote it by $\boldsymbol{x}(\mu_k)$, by using ε_k as a stopping tolerance. Let

$$\phi_{k+1}(t) = \sum_{j=1}^{n} f_j(t) x_j(\mu_k) - g(t), \quad \forall t \in T.$$

Find a minimizer t_{k+1} of $\phi_{k+1}(t)$ over T and calculate

$$\phi_{k+1}(t_{k+1}) = \sum_{j=1}^{n} f_j(t_{k+1})\, x_j(\mu_k) - g(t_{k+1}).$$

Step 3: If $\phi_{k+1}(t_{k+1}) \geqslant -\eta_3$, then check μ_k ad ε_k.
If $\mu_k \leqslant \eta_1$ and $\varepsilon_k \leqslant \eta_2$, then Stop! $\boldsymbol{x}(\mu_k)$ is optimal to LSIP.
Otherwise, Go to Step 4.
If $\phi_{k+1}(t_{k+1}) < -\eta_3$, Go to Step 4. ($\boldsymbol{x}(\mu_k)$ is not feasible to LSIP.)
Step 4: Let $T_{k+1} = T_k \cup \{t_{k+1}\}$.
Update μ_k: if $\mu_k > \eta_4$, then set $\mu_{k+1} = \delta_1 \mu_k$; otherwise, set $\mu_{k+1} = \mu_k$.
Update ε_k: if $\varepsilon_k > \eta_2$, then set $\varepsilon_{k+1} = \delta_2 \varepsilon_k$; otherwise, set $\varepsilon_{k+1} = \varepsilon_k$.
Reset $k \leftarrow k + 1$ and Go to Step 2.

The reduction of ε_k creates a "cone" in the solution space with its vertex representing the ending iteration; whereas the exact approach causes a "tube" since the tolerance is independent of k. With

this ε_k-inexactness, Algorithm 2 seeks only approximate solutions of LP_{μ_k} at the beginning of the framework. At least in theory, this could further reduce computational work. In case we choose $\varepsilon_k = \eta_2$, for all k, Algorithm 2 is reduced to Algorithm 1 whose convergence result is shown in [9]. Moreover, once a fixed $\eta_2 > 0$ is given, ε_k will be geometrically reduced to a number which is smaller than the *fixed* η_2 in finite steps. Thus, if we regard ε_k-inexactness as an implementation technique, then the convergence proof of Theorem 1 in [9] is also valid for Algorithm 2.

3. Implementation issues

The objective of this paper is to perform numerical experiments to confirm that the inexact approach (Algorithm 1) and its extension (Algorithm 2) are potentially superior to the exact approach. In Section 4, we shall study four different methods for solving LSIP with the help of four sets of examples; here we discuss some implementation issues in this section. The first method, denoted by "Inexact1", is an implementation of Algorithm 1. The second method, "Inexact2", is an implementation of Algorithm 2. The third method, "Exact" method, is the traditional exact approach of [1, 12, 13, 16]. Its solution framework is the same as Algorithm 1, but the perturbation parameters μ_k are taken to be zero for all k. For the details of the Exact method, *the same four-step outline of Algorithm* 1 will be referred. In this way, we solve a linear program LP_k with k linear constraints in Step 2 by using a small tolerance η_2; and its optimal solution is denoted by x^k. Since there is no inexactness involved, we call it an "exact" approach. The fourth method, "Discretized" method, is to discretize the domain T into $p > 0$ points and then solve a linear program with p linear constraints evaluated at these points. This method is only a finite approximation of LSIP. In general, more points result in better approximation but with higher computational burden. The commercial solver MINOS [18] with its default parameters is used to solve the linear programs occurred in the Discretized method. The numbers of grid points p are chosen to be 100, 250, 400, 550. However, only $p = 550$ is reported unless the solution accuracies with different p-values are significantly different. The solutions of this method are used for references only.

In Step 2 of Algorithms 1 and 2, although, in theory, several methods for solving entropic optimization problems [14, 19] can also be used, we adopt the dual method developed in [7] for our implementation. We briefly describe some features of this dual method. Given $\mu_k > 0$, the solution $x(\mu_k)$ of LP_{μ_k} can be found via solving an *unconstrained* convex dual program. The authors denoted this dual by $(\mathrm{D}\mu_k)$ and its variables by $w(\mu_k)$. The curved search method of [2] was adopted in [7] to achieve a result of global convergence with a quadratic rate of convergence for the dual method. Moreover, it was proven in [7] that the solution of LP_{μ_k} as an "ε-optimal" solution of LP_k, for any given $\varepsilon > 0$, when μ_k is chosen to be sufficiently small. This further explains why $x(\mu_k)$ is a good approximate solution of LP_k, when μ_k is sufficiently small.

In our implementation, $t_1 = 0.5$ is always chosen as a starting point for the Inexact1, Inexact2, and Exact methods. For the Inexact1 and Inexact2 methods, we set $\eta_1 = 10^{-3}$, $\eta_2 = 10^{-8}$, $\eta_3 = 10^{-4}$, $\eta_4 = 10^{-5}$, and

$$\mu_1 = 0.1,$$
$$\mu_{k+1} = \begin{cases} \delta_1 \mu_k & \text{if } \mu_k > 10^{-5}, \\ \mu_k & \text{otherwise,} \end{cases} \tag{3.1}$$

where δ_1 in (3.1) will be specified later. According to [7], μ_k need not be very small *numerically* to assure that $x(\mu_k)$ is a good approximation of x^k. Therefore, we use the same control parameters to run the code of the dual algorithm developed in [7]. To stop the dual algorithm, a tolerance level in optimality condition is set to be $\eta_2 = 10^{-8}$ for the Inexact1 method. For the Inexact2 method, the tolerance is set to be ε_k according to the following formula:

$$\varepsilon_1 = 0.1,$$

$$\varepsilon_{k+1} = \begin{cases} \delta_2 \varepsilon_k & \text{if } \varepsilon_k > 10^{-8}, \\ \varepsilon_k & \text{otherwise.} \end{cases} \tag{3.2}$$

In the Exact method, η_3 is also set to be 10^{-4} and we use ALPO, an interior point LP solver provided in [20], to solve LP_k in each iteration. No μ_k- or ε_k-inexactness is added and the stopping rule for each LP_k is set by specifying the duality gap to be $\leqslant \eta_2 = 10^{-6}$. All other default values of ALPO are kept unchanged. It is well known that the major computational effort for solving LP_k is determined by the number of Cholesky factorizations ($k \times k$, in size) required.

To optimize the function ϕ_{k+1} over set T, special properties of ϕ_{k+1} should be taken into consideration for efficient computation. However, since our main objective is to compare the performance of the Inexact1, Inexact2, and Exact methods, as long as we use the same mechanism in all cases, there is no urgent need to use a more sophisticated code. In our implementation, since T is a one-dimensional compact set in all testing problems, we simply partition T into q equally-distant points and select the minimizer of $\phi_{k+1}(t)$, over these finite number of grid points q, as an approximate solution of the true minimizer of $\phi_{k+1}(t)$ in the domain T. In the first five iterations, q is set to be 10^4, and 10^5 thereafter. Let $T_k = \{t_1, t_2, \ldots, t_k\}$ be considered as a partition in T. In case t_{k+1} is too close to t_k (we call this "trapping"), we reset t_{k+1} to be the midpoint of the largest subinterval of the current partition and then continue.

In order to run the dual solver of [7], free variables are converted to nonnegative variables by letting $x = x^+ - x^-$ with $x^+, x^- \geqslant 0$. With finite arithmetics, the entropic dual solvers could encounter numerical difficulties in overflow, if μ_k is too small and the initial solution of the unconstrained dual $\mathrm{D}\mu_k$ is far away from the feasible region of the dual linear program of LP_k (see [7]). To overcome this, a "warm start" is selected for the Inexact1 and Inexact2 methods by setting the new component of the dual variable $w(\mu_k)_k = 0$ and using the optimal solution $w(\mu_{k-1})$ of the previous subproblem $\mathrm{D}\mu_{k-1}$ as the first $k-1$ components of the initial solution for $\mathrm{D}\mu_k$.

When Algorithm 1 or 2 terminates, we let $\bar{k}$ be the index of the last iteration and report $x^* = x(\mu_{\bar{k}})$ as an optimal solution of LSIP. Recall that, by definition, we have $\phi_{\bar{k}+1}(t_{\bar{k}+1}) = \text{Minimum } \sum_{j=1}^{n} f_j(t) x_j^* - g(t)$. This value indicates the "primal feasibility". If $\phi_{\bar{k}+1}(t_{\bar{k}+1}) \geqslant 0$, the final solution is primal feasible. Otherwise, $\phi_{\bar{k}+1}(t_{\bar{k}+1}) < 0$ is reported as the "primal infeasibility". Also note that, when the algorithm terminates, an optimal dual solution w^* of $\mathrm{LP}_{\mu_{\bar{k}}}$ is proved by the dual solver of [7]. Checking the components of w^*, an approximate optimal solution of DLSIP can be identified. Then the "dual objective value" and "dual infeasibility" can be calculated accordingly and reported. Since the final optimal primal and dual solutions obtained in this way are all approximate solutions, we refer to the *absolute value* of the difference between the primal objective value and dual objective value as the "duality gap".

4. Numerical results

4.1. L1 problems

In this section, we compare the numerical performance of different methods by testing some commonly encountered L1 examples [12, 10] with simple modifications.

Problem 1.

$$\min \quad \sum_{j=1}^{n} \frac{x_j}{j}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} t^{j-1} x_j \geqslant \frac{1}{2-t}, \quad t \in [0, 1],$$

$$x_j \geqslant 0, \quad j = 1, 2, \ldots, n.$$

Problem 2.

$$\min \quad \sum_{j=1}^{n} \frac{x_j}{j}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} t^{j-1} x_j \geqslant \sum_{j=0}^{4} t^{2j}, \quad t \in [0, 1],$$

$$x_j \geqslant 0, \quad j = 1, 2, \ldots, n.$$

Problem 3.

$$\min \quad \sum_{j=1}^{n} \frac{x_j}{j}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} t^{j-1} x_j \geqslant e^t, \quad t \in [0, 1],$$

$$x_j \geqslant 0, \quad j = 1, 2, \ldots, n.$$

By varying the value of n, each testing problem actually represents a set of problems. We have tested different size problems and chose $n = 50$ as a baseline case for illustration purposes.

4.1.1. Testing with nonnegative variables

Notice that Problems 1–3 have nonnegative variables. In this section, we use the testing results of $n = 50$ to separate the effect of allowing inexactness in μ_k from that of inexactness of ε_k. With few exceptions, the unreported cases with different n gave similar results.

The parameters for reduction ratios were set to be $\delta_1 = \frac{1}{4}$ for both Algorithms 1 and 2 and $\delta_2 = \frac{1}{8}$ for Algorithm 2. Table 1 shows the test results for Problem 1.

Table 1
Test results for Problem 1

Iteration	Number of Cholesky fatorizations			Points	Discretized objective value
	Inexact1	Inexact2	Exact		
1	5	2	12	550	$6.931488507 \cdot 10^{-1}$
2	4	3	12		
3	16	7	13		
4	9	9	13		
5	10	9	14		
6	15	15	19		
7			18		
Total	59	45	101		
Primal obj. value	$6.933146121 \cdot 10^{-1}$	$6.931884209 \cdot 10^{-1}$	$6.931473308 \cdot 10^{-1}$		
Dual obj. value	$6.930959336 \cdot 10^{-1}$	$6.930971924 \cdot 10^{-1}$			
Duality gap	$2.186784836 \cdot 10^{-4}$	$9.122854717 \cdot 10^{-5}$			
Primal infeasibility	$4.028447122 \cdot 10^{-5}$	$-8.627405574 \cdot 10^{-5}$			
Dual infeasibility	$2.999972688 \cdot 10^{-5}$	$2.997638212 \cdot 10^{-5}$			

Table 1 shows that both inexact methods took six iterations while the Exact method took seven iterations to reach an "optimal" solution of this LSIP whose optimal objective value is very close to the value obtained by the Discretized method with 550 grid points. In the first iteration, Inexact1 took five Cholesky factorizations, Inexact2 took two, while the Exact method required 12 Cholesky factorizations of the same size and structure. Similarly, in the second iteration, the three methods required four, three and 12 factorizations respectively. In total, Inexact1 took 59 Cholesky factorizations, Inexact2 took 45, while the Exact method required 101 factorizations. The dual information shown in the table also supported the quality of the "optimal solution" we obtained.

According to our experience, with very few exceptions, the numbers of iterations required by Inexact1, Inexact2 and Exact methods are very close for problems with nonnegative variables. The major difference was in the numbers of Cholesky factorizations required at each iteration. Both the inexact methods required much less computation at the early iterations to reach an approximate solution than the exact method. Surprisingly, the approximate solutions worked very well and the inexact method terminated earlier than the exact method for this testing problem. This validates the potential of the inexact approach.

Also notice that Inexact2 required fewer number of Cholesky factorizations than Inexact1 at each iteration. This could be considered as the contribution of the additional inexactness in ε_k, which was the only difference between Inexact2 and Inexact1 methods.

Test results of Problem 2 are given in Table 2.

These results further confirmed that the inexact approach is a good way to reduce computational work. However, "too much inexactness" like Inexact2 could cause a "not so good" approximate solution which eventually required Inexact2 to take extra number of iterations to reach optimality. In this case, Inexact1 could do better.

Test results of Problem 3 are listed in Table 3.

The results shown in Table 3 are very similar to those of Problem 1.

4.1.2. Testing with free variables

We have concluded in Section 4.1.1 that the inexact approach of [9] indeed outperformed the traditional exact approach in our experiments. Since the Inexact2 method involved two-layer inexactness for less computational requirement and never encountered overflow problem in our previous tests, we now focus on it for further tests with free variables. In order to show that Algorithm 2 is not very sensitive to the choice of reduction ratios, we reset them to a set of larger values ($\delta_1 = \delta_2 = \frac{1}{2}$) and rename it as "*Inexact method*", instead of Inexact2.

The testing problems listed below are very close to Problems 1 and 2.

Problem 1′.

$$\min \quad \sum_{j=1}^{n} \frac{x_j}{j}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} t^{j-1} x_j \geqslant \frac{1}{2-t}, \quad t \in [0, 1].$$

Table 2
Test results for Problem 2

Iteration	Number of Cholesky factorizations			Points	Discretized objective value
	Inexact1	Inexact2	Exact		
1	2	2	11	550	1.787347451
2	6	4	13		
3	6	6	15		
4	6	6	12		
5	8	8	18		
6	9	8	18		
7	22	23	18		
8	11	17	22		
9		3			
10		10			
11		4			
12		10			
Total	70	101	127		
Primal obj. value	1.787309785	1.787345538	1.787298883		
Dual obj. value	1.787319867	1.787321774			
Duality gap	$1.008201860 \cdot 10^{-5}$	$2.376375898 \cdot 10^{-5}$			
Primal infeasibility	$-4.461871286 \cdot 10^{-5}$	$1.130492955 \cdot 10^{-5}$			
Dual infeasibility	$6.103457933 \cdot 10^{-6}$	$6.103585204 \cdot 10^{-6}$			

Table 3
Test results for Problem 3

Iteration	Number of Cholesky factorizations			Points	Discretized objective value
	Inexact1	Inexact2	Exact		
1	2	2	11	550	1.718323395
2	10	4	13		
3	9	6	13		
4	13	8	13		
5	14	14	14		
6			16		
Total	48	34	80		
Primal obj. value	1.718260027	1.718245760	1.718266823		
Dual obj. value	1.719019923	1.719019912			
Duality gap	$7.598959569 \cdot 10^{-4}$	$7.741521829 \cdot 10^{-4}$			
Primal infeasibility	$-3.771960315 \cdot 10^{-5}$	$-6.382805464 \cdot 10^{-5}$			
Dual infeasibility	$3.906508305 \cdot 10^{-4}$	$3.906493766 \cdot 10^{-4}$			

Problem 2′.

$$\min \quad \sum_{j=1}^{n} \frac{x_j}{j}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} t^{j-1} x_j \geqslant -\sum_{j=0}^{4} t^2 j, \quad t \in [0, 1].$$

Note that Problem 1′ is obtained from Problem 1 by relaxing the nonnegativity requirement on its variables. For the case $n = 8$, the problem has been studied in [12, p. 137]. Problem 2′ is also closely related to Problem 2. For $n = 7$, it becomes a problem studied in [10]. Test results are presented in the Tables 4–7.

The results further confirmed that, in general, the inexact approach is more efficient than the exact approach. In our test, when $n = 50$, ALPO for the Exact method reported an instability warning of "suboptimal solution found" for both Problems 1′ and 2′ and the Exact method required more than 50 iterations. However, for a small size problem like Case 1 of Problem 2′ with relatively large values of δ_1 and δ_2 (which imply slow reductions in inexactness parameters μ_k and ε_k), "too much inexactness" could defeat its own purpose. In summary, for both types of L1 problems, the Inexact method has been shown to be more efficient than the Exact method. We shall further exploit the uniform approximation problem in the next subsection.

Table 4
Test results for Problem 1′; Case 1: $n = 8$

Iteration	Number of Cholesky factorizations		Points	Discretized objective value
	Inexact	Exact		
1	2	6	550	$6.931481715 \cdot 10^{-1}$
2	3	6		
3	3	6		
4	4	8		
5	6	10		
6	8	11		
7	11	14		
8		14		
Total	37	75		
Primal obj. value	$6.930961577 \cdot 10^{-1}$	$6.931476253 \cdot 10^{-1}$		
Dual obj. value	$6.938219015 \cdot 10^{-1}$			
Duality gap	$7.257438375 \cdot 10^{-4}$			
Primal infeasibility	$-9.130189038 \cdot 10^{-5}$			
Dual infeasibility	$9.934666969 \cdot 10^{-4}$			

Table 5
Test results for Problem 1'; Case 2: $n = 50$

Iteration	Number of Cholesky factorizations		Points	Discretized objective value
	Inexact	Exact		
1	2		[b]	
2	3			
3	4			
4	5			
5	5			
6	5			
7	5			
8	8			
9	8			
10	8			
11	10			
12	10			
13	13			
Total	86	[a]		
Primal obj. value	$6.934730174 \cdot 10^{-1}$	$6.928672035 \cdot 10^{-1}$		
Dual obj. value	$6.931576713 \cdot 10^{-1}$			
Duality gap	$3.153461100 \cdot 10^{-4}$			
Primal infeasibility	$1.775384269 \cdot 10^{-4}$			
Dual infeasibility	$1.552854735 \cdot 10^{-5}$			

[a] Exact method took more than 50 iterations and the reported value was the objective value of LP_{50}.
[b] MINOS failed due to insufficient memory space or bad scaling.

4.2. *Uniform approximation problem*

The uniform approximation problem was studied in [10]. We found it interesting and successfully solved the same problem by using a fifth degree polynomial to approximate x^6. However, since its optimal value was very close to zero, the numerical "relative error" could be too large for solution accuracy in higher dimensional problems. Thus we tailored the problem with a sine function in test Problem 4. More about the uniform approximation problem can be found in [1, Ch. 4].

Problem 4.

$$\min_{x \in \mathbb{R}^n} \max_{t \in [0,1]} | \sin(150t) - \sum_{j=1}^{n} x_j t^{j-1} |.$$

This is equivalent to the following LSIP:

Table 6
Test results for Problem 2'; Case 1: $n = 7$

Iteration	Number of Cholesky factorizations		Points	Discretized objective value
	Inexact	Exact		
1	2	5	550	−1.786830557
2	4	5		
3	4	7		
4	5	11		
5	6	14		
6	6	16		
7	13	21		
8	9	19		
9	8	23		
10	13	10		
11	17	10		
12	17			
13	17			
14	18			
Total	139	141		
Primal obj. value	−1.786569473	−1.786906749		
Dual obj. value	−1.785850451			
Duality gap	$7.190226946 \cdot 10^{-4}$			
Primal infeasibility	$-8.460290098 \cdot 10^{-6}$			
Dual infeasibility	$3.934195879 \cdot 10^{-5}$			

$$\text{Minimize} \quad h$$

$$\text{s.t.} \quad h + \sum_{j=1}^{n} t^{j-1} x_j \geqslant \sin(150t),$$

$$h - \sum_{j=1}^{n} t^{j-1} x_j \geqslant -\sin(150t), \quad t \in [0, 1].$$

Note that this is also a problem with free variables. We set $n = 10$ and $n = 20$ for testing purposes. The results are shown in Tables 8 and 9.

It is easy to see that the optimal objective value of Problem 4 is 1, in both the cases. Therefore, the infeasibility issue was not further pursued and reported in Tables 8 and 9. It is also interesting to observe that, in Case 1, both the inexact and exact methods converged very closely to the same optimal solution, while the solutions obtained by the Discretized method varied depending upon the number of grid points used. The reason could be the oscillation of $\sin(150t)$ which caused too many local maxima and minima in the domain of $T = [0, 1]$. Therefore the result from the Discretized method depended heavily on the grid points used. This case clearly demonstrated the

Table 7
Test results for Problem 2'; Case 2: $n = 50$

Iteration	Number of Cholesky factorizations		Points	Discretized objective value
	Inexact	Exact		
1	2		[b]	
2	3			
3	4			
4	5			
5	5			
6	6			
7	5			
8	7			
9	9			
10	8			
11	10			
12	11			
13	17			
14	14			
Total	106	[a]		
Primal obj. value	-1.787262155	-1.787934148		
Dual obj. value	-1.787251316			
Duality gap	$1.083840683 \cdot 10^{-5}$			
Primal infeasibility	$-4.798607474 \cdot 10^{-5}$			
Dual infeasibility	$1.359835229 \cdot 10^{-5}$			

[a] Exact method took more than 50 iterations and the reported value was the objective value of LP_{50}.
[b] MINOS failed due to insufficient memory space or bad scaling.

Table 8
Test results for Problem 4; Case 1: $n = 10$

Iteration	Number of Cholesky factorizations		Points	Discretized objective value
	Inexact	Exact		
1	3		100	$9.955884409 \cdot 10^{-1}$
2	5		250	$9.970603354 \cdot 10^{-1}$
3	6		400	$9.996166268 \cdot 10^{-1}$
4	6		550	$9.976483775 \cdot 10^{-1}$
5	5			
6	6			
7	9			
8	11			
9	9			
10	11			
Total	71	574[a]		
Primal obj. value	$9.999930569 \cdot 10^{-1}$	$9.999958223 \cdot 10^{-1}$		
Dual obj. value	1.006542646			
Duality gap	$6.549589998 \cdot 10^{-3}$			

[a] The Exact method took 39 iterations to reach this solution.

Table 9
Test results for Problem 4; Case 2: $n = 20$

Iteration	Number of Cholesky factorizations		Points	Discretized objective value
	Inexact	Exact		
1	3		100	$9.936574538 \cdot 10^{-1}$
2	5		250	$9.969951180 \cdot 10^{-1}$
3	6		400	$9.988890651 \cdot 10^{-1}$
4	7		550	$9.975850019 \cdot 10^{-1}$
5	8			
6	6			
7	8			
8	13			
9	14			
10	9			
11	12			
12	9			
13	18			
14	13			
15	13			
16	8			
17	9			
18	16			
19	9			
20	30			
Total	216	[a]		
Primal obj. value	$9.999998555 \cdot 10^{-1}$	$9.979263571 \cdot 10^{-1}$		
Dual ob. value	1.000411514e			
Duality gap	$4.116585775 \cdot 10^{-4}$			

[a] The Exact method requires more than 50 iterations and the reported value is the objective value of LP_{50}.

robustness and efficiency of the Inexact method. Case 2 showed similar results. It is worth mentioning that the Exact method encountered "trapping" problems (as described in Section 3) many times, but the Inexact method did not encounter this problem. One explanation is that two consecutive subproblems for the Inexact method differ more (due to different μ_k and ε_k) than those in the Exact method.

5. Conclusion

In this paper, we have validated numerically that the inexact approach proposed in [9] is robust and computationally efficient. We have also extended the original algorithm by including one more layer of inexactness. This, in general, further reduces computational work, even with quite arbitrary

choices of the reduction ratio parameters. However, for some cases of small size problems with large reduction ratios, "too much inexactness" may become undesirable.

Acknowledgements

The authors would like to thank Dr. H.-S. Jacob Tsao for his help in conducting test runs. Without his codes, this work would not be complete.

References

[1] E.J. Anderson and P. Nash, *Linear Programming in Infinite-dimensional Spaces* (Wiley, Chichester, 1987).
[2] A. Ben-Tal, A. Melman and J. Zowe, Curved search methods for unconstrained optimization, *Optimization* **21** (1990) 669–695.
[3] J.R. Erickson, Algorithm for entropy and mathematical programming, Ph.D. Thesis, Linköping University, Linköping, Sweden, 1981.
[4] J.R. Erickson, Using the entropy function to get interior point methods for mathematical programming, Technical Report LiTH-MAR-R-1990-02, Linköping Univ., Linköping, Sweden, 1990.
[5] S. Erlander, Entropy in linear programs, *Math. Programming* **21** (1981) 137–151.
[6] S.C. Fang, An unconstrained convex programming view to linear programming, *Z. Oper. Res.*, **36** (1992) 149–161.
[7] S.C. Fang and H.S.J. Tsao, Linear programming with entropic perturbation, *Z. Oper. Res.* **37** (1993) 171–186.
[8] S.C. Fang and S.Y. Wu, Entropic path-following for linear semi-infinite programming, *Mathematics Today* **XII-A** (1994) 1–12.
[9] S.C. Fang and S.Y. Wu, An inexact approach to solving linear semi-infinite programming problems, *Optimization* **28** (1994) 291–299.
[10] M.C. Ferris and A.B. Philpott, An interior point algorithm for semi-infinite linear programming, *Math. Programming* **43** (1989) 257–276.
[11] M.C. Ferris and A.B. Philpott, On affine scaling and semi-infinite programming, *Math. Programming* **56** (1992) 361–364.
[12] S.Å. Gustafson and K. Glashoff, *Linear Optimization and Approximation* (Springer, New York, 1982).
[13] S.Å. Gustafson and K. Kortanek, Numerical treatment of a class of semi-infinite programming problems, *Naval Res. Logist. Quarter.* **20** (1973) 473–504.
[14] C.G. Han, P.M. Pardalos and Y. Ye, Implementation of interior-point algorithms for some entropy optimization problems, *Optim. Methods Software* **1** (1992) 71–80.
[15] N. Karmarkar, A new polynomial time algorithm for linear programming, *Combinatorica* **4** (1984) 373–395.
[16] H.C. Lai and S.Y. Wu, On linear semi-infinite programming problems, an algorithm, *Numer. Funct. Anal. Optim.* **13** (1992) 287–304.
[17] H.C. Lai and S.Y. Wu, Linear programming in measure spaces with an algorithm, accepted for publication in *Optimization.*
[18] B.A. Murtagh and M.A. Saunders, MINOS 5.1 user's guide, Report SOL 83-20R, Dept of Operations Research, Stanford Univ., Stanford, CA, 1987.
[19] F. Potra and Y. Ye, An algorithm for solving entropy optimization problems with globally linear and locally quadratic convergence, *SIAM J. Optim.* **3** (1994) 843–860.
[20] R.J. Vanderbei, ALPO: Another linear program solver, AT & T Bell Laboratories, Murray Hill, NJ, 1990; Dept. of Civil Engineering and Operations Research, Princeton Univ., Princeton, NJ, revised in 1992.