On the Worst Case Complexity of Potential Reduction Algorithms for Linear Programming

Dimitris Bertsimas
and
Xiaodong Luo

WP# -3558-93 MSA April, 1993

On the Worst Case Complexity of Potential Reduction Algorithms for Linear Programming

Dimitris Bertsimas [1] Xiaodong Luo [2]

April 1993

[1]Dimitris Bertsimas, Sloan School of Management and Operations Research Center, MIT, E53-359, Cambridge, MA 02139. The research of the author was partially supported by a Presidential Young Investigator Award DDM-9158118 with matching funds from Draper Laboratory.
[2]Xiaodong Luo, Operations Research Center, MIT, Cambridge, MA 02139.

Abstract

There are several classes of interior point algorithms that solve linear programming problems in $O(\sqrt{n}\ L)$ iterations, but it is not known whether this bound is tight for any interior point algorithm. Among interior point algorithms, several potential reduction algorithms combine both theoretical ($O(\sqrt{n}\ L)$ iterations) and practical efficiency as they allow the flexibility of line searches in the potential function and thus can lead to practical implementations. It is a significant open question whether interior point algorithms can lead to better complexity bounds. In the present paper we give some negative answers to this question for the class of potential reduction algorithms. We show that, without line searches in the potential function, the bound $O(\sqrt{n}\ L)$ is tight for several potential reduction algorithms, i.e., there is a class of examples, in which the algorithms need at least $\Omega(\sqrt{n}\ L)$ iterations to find an optimal solution. In addition, we show that for a class of potential functions, even if we allow line searches in the potential function, the bounds are still tight. We note that it is the first time that tight bounds are obtained for any interior point algorithm.

1 Introduction

In the last decade there has been a lot of excitement about interior point algorithms for linear programming (LP). In his seminal work, Karmarkar [7] presented an $O(nL)$ iterations algorithm with $O(n^3)$ work per iteration thus resulting in an $O(n^4L)$ algorithm for the linear programming problem (LP):

$$\begin{aligned} \textit{minimize} \quad & c'x \\ \textit{subject to} \quad & Ax = b \\ & x \geq 0, \end{aligned}$$

where A is an $m \times n$ matrix and L is the input size of the linear program. Moreover, he used rank one updates to obtain an $O(n^{3.5}L)$ algorithm for LP. Since then, there have been several path following algorithms for LP that need $O(\sqrt{n}L)$ iterations (Renegar [10], Gonzaga [4]) with time complexity $O(n^3L)$. The drawback of the latter approaches is that, although they achieve the best known worst case bound, they are restricted to make small steps by following the central trajectory. As a result, these path following methods are not attractive for a practical implementation.

A major step in the direction of practical algorithms, yet achieving the best known worst case bounds, was taken by Ye [12], in which he presented a potential reduction primal-dual algorithm, which solves linear programming in $O(\sqrt{n}L)$ iterations and $O(n^3L)$ operations. Freund [3] simplified the algorithm by Ye and showed that certain choices for the parameters in the algorithm are in a sense best possible. Gonzaga and Todd [5] proposed another potential reduction algorithm, which also solves linear programming in $O(\sqrt{n}L)$ iterations. The major advantage of this class of algorithms is that they are conceptually elegant and natural, achieve the best known worst case bounds and yet they lead to practical implementations through the use of line searches. In the theory and practice of linear programming this is a rare combination. The simplex method and affine scaling algorithm for example are efficient practical algorithms, but are not known to be polynomial algorithms.

Since a variety of interior point algorithms solve LP in $O(\sqrt{n}L)$ iterations, one can naturally ask:

1. *Can the bound $O(\sqrt{n}L)$ be improved for any interior point algorithm, i.e., is the bound inherent in the algorithm or is it because of the weakness of our proof methods ?*

2. *Do line searches improve the theoretical complexity of interior point algorithms ?*

3. *More ambitiously, is the bound $O(\sqrt{n}L)$ inherent in the linear programming problem ? We remark that this question is rather deep as it is related with the existence of a strongly polynomial algorithm for LP.*

Related with the first two questions Anstreicher [1] (see also McDiarmid [8]) showed that in the worst-case, there is a step among all steps of Karmarkar's algorithm such that the reduction of the potential function can not be greater than a constant. Anstreicher [2] sharpened this observation and showed that with exact line searches, there is a class of linear programs, in which Karmarkar's algorithm can only obtain a decrease of the potential function that is no more than a constant for every iteration; he further showed that $\Theta(\log n\ L)$ iterations are actually needed to solve this example. Finally, Kaliski and Ye [6] showed that Karmarkar's algorithm requires at least $\Omega(n)$ iterations to converge for solving a linear program with only one constraint. Although the above results provide insights, they do not answer the first two questions.

In the present paper we give negative answers to the first two questions for the potential reduction algorithms of Ye [12], Freund [3], and Gonzaga and Todd [5]. We show that, without line searches in the potential function, the bound $O(\sqrt{n}\ L)$ is tight, i.e., there is a class of examples, in which the algorithms need at least $\Theta(\sqrt{n}\ L)$ iterations to find an optimal solution. In addition, we show that even if we allow line searches in the potential function in Ye's algorithm, the bound is still tight. We note that it is the first time that tight bounds are obtained for any interior point algorithm.

The paper is structured as follows. In Section 2 we summarize the potential reduction algorithm and the known complexity results. In Section 3 we exhibit the example that achieves $\Theta(\sqrt{n}\,L)$ iterations without line searches, while in Section 4 we exhibit the example that achieves $\Theta(\sqrt{n}\ L)$ iterations even if we allow line searches. The final section contains some observations, critical comments, conjectures and open problems.

We briefly describe our notation below. All vectors are column vectors and the superscript $'$ denotes transpose. Unless otherwise specified, $\|\cdot\|$ denotes the usual Euclidean norm, $\log(\cdot)$ denotes the natural logarithmic function. For any $x \in \Re^n$, we denote by x_i the i–th coordinate of x, and when describing an algorithm, we use the notation x^k to denote the value of variable x evaluated at the k–th iteration of the algorithm.

2 The Potential Reduction Algorithm

We are interested in solving the linear programming problem

$$\begin{aligned} (P) \quad \textit{minimize} \quad & c'x \\ \textit{subject to} \quad & Ax = b \\ & x \geq 0, \end{aligned}$$

and its dual

$$\begin{aligned} (D) \quad \textit{maximize} \quad & b'x \\ \textit{subject to} \quad & A'y + s = c \\ & s \geq 0. \end{aligned}$$

Todd and Ye [11] introduced the primal-dual logarithmic potential function:

$$G(x,s) = q \ \log \ x's - \sum_{j=1}^{n} \log \ x_j - \sum_{j=1}^{n} \log \ s_j, \tag{1}$$

where the first term is a measure of the duality gap $x's$ and the two other terms are the barrier functions. The goal of the algorithm proposed by Ye [12] is to decrease the duality gap below a tolerance ϵ by decreasing the potential function $G(x,s)$ at every iteration. We first give the following definition.

Definition 1 *A primal and dual solution pair (x,s) is called ϵ-optimal if the duality gap satisfies $x's \leq \epsilon$.*

Let L be the input size of the linear program (P), i.e., we use the definition[1]

$$L \triangleq size(det_{\max}) + size(b_{\max}) + size(c_{\max}) + size(m+n)$$

where

$$\begin{aligned} det_{\max} &\triangleq \max_{A_1}(|\det(A_1)|) \\ b_{\max} &\triangleq \max_{i}(|b_i|) \\ c_{\max} &\triangleq \max_{j}(|c_j|) \end{aligned}$$

[1] In the literature, there are many definitions for L, see for example [9] for a different definition. This definition of L, however, suffices to prove the usual rounding theorem that if an ϵ-optimal solution is available with $\epsilon = 2^{-2L}$, then an exact solution can be found in polynomial time through rounding. More importantly, the value of L defined here is always smaller than in [9].

A_1 is any square submatrix of A and $size(\cdot)$ is the number of binary bits that are needed to represent the value of its argument. It is well known that when an ϵ-optimal solution is available with $\epsilon = 2^{-2L}$, then an exact solution can be found in polynomial time through rounding.

For this class of potential functions the following result holds:

Theorem 1 [Ye [12]] *An algorithm that reduces $G(x,s)$ by an amount greater or equal to $\delta > 0$ at each iteration finds an ϵ-optimal solution $(\bar{x}, \bar{s})$ in $O(\frac{q-n}{\delta} \log \frac{1}{\epsilon})$ iterations, as long as the initial feasible primal-dual values (x^0, s^0) satisfy $G(x^0, s^0) \leq O(\frac{q-n}{\delta} \log \frac{1}{\epsilon})$.* □

We note that the previous theorem focuses to finding an ϵ-optimal solution. If we want an exact solution we need to replace $\log \frac{1}{\epsilon}$ by L.

Let (A, b, c) be the data for the LP (P); let $x^0 > 0, s^0 > 0, y^0$ be the initial primal and dual solutions respectively; let $\epsilon > 0$ be the optimality tolerance. Let X denote a diagonal matrix with $(x_1, x_2, \ldots, x_n)$ in the diagonal. Let e_n be a column vector of all ones. Then Ye's potential function algorithm which uses the parameters (γ, α, q) can be described as follows:

Algorithm $\mathcal{A}$ $(A, b, c, x^0, s^0, y^0, \epsilon)$.

$k = 0$;

while $(x^k)'s^k > \epsilon$ do

$X^k = diag(x^k)$;

$\bar{A}^k = (AX^k)'(AX^kX^kA')^{-1}AX^k$;

$u^k = (I - \bar{A}^k)(\frac{q}{(x^k)'s^k} X^k s^k - e_n)$;

if $||u^k|| \geq \gamma$

then *(primal step)*

$x^{k+1} = x^k - \alpha \frac{X^k u^k}{||u^k||}$;

$s^{k+1} = s^k$;

else *(dual step)*

$x^{k+1} = x^k$;

$s^{k+1} = \frac{(x^k)'s^k}{q} (X^k)^{-1}(u^k + e_n)$;

end if

$k = k + 1$;

end while

If $||u^k|| \geq \gamma$ we say that the algorithm performs *a primal step*, while if $||u^k|| < \gamma$, we say that the algorithm performs *a dual step*. The algorithm proposed by Freund is somewhat different in the dual step: It finds a scalar $\Delta^k \in (0, (x^k)'s^k]$ such that

$$||(I - \bar{A}^k)(\frac{q}{\Delta^k} X^k s^k - e_n)|| = \gamma$$

and then updates the dual slacks as follows:

$$\bar{u}^k = (I - \bar{A}^k)(\frac{q}{\Delta^k} X^k s^k - e_n),$$

$$s^{k+1} = \frac{(x^k)'s^k}{q} (X^k)^{-1}(\bar{u}^k + e_n).$$

Moreover, Gonzaga and Todd's [5] algorithm uses exactly the same primal step, while the dual step is more symmetric than both Ye's and Freund's algorithms. The algorithm uses the projected gradients of the potential function both in the primal and in the dual space.

We also remark that we can improve the performance of all these algorithms by using line searches in the potential function, i.e., by varying α to maximize the decrease of the potential function at each step rather than keeping it fixed throughout the algorithm.

For Algorithm $\mathcal{A}$ the following theorem holds:

Theorem 2 [Freund [3], Ye [12]]

1. *If Algorithm $\mathcal{A}$ executes a primal step at step k and the step size is chosen to be α, then*

$$G(x^{k+1}, s^{k+1}) - G(x^k, s^k) \leq -\alpha\gamma + \frac{\alpha^2}{2(1-\alpha)}.$$

If the algorithm executes a dual step at step k then

$$G(x^{k+1}, s^{k+1}) - G(x^k, s^k) \leq -\frac{q-n}{q}(q - n - \gamma\sqrt{n}) + \frac{2\gamma^2}{(1-\gamma)(1-3\gamma)}.$$

2. *If $q = n + \sqrt{n}$, $\gamma = 0.22$, $\alpha = \frac{1}{6}$, then Algorithm $\mathcal{A}$ reduces $G(x, s)$ by a constant amount $\delta = 0.02$ at each iteration, and thus finds an ϵ-optimal solution in $O(\sqrt{n}\log\frac{1}{\epsilon})$ iterations, provided that $G(x^0, s^0) = O(\sqrt{n}\log\frac{1}{\epsilon})$.*

We notice that in order to minimize the number of iterations we need to choose parameters (α, γ) so that:

$$\min_{\alpha,\gamma} \frac{q-n}{\min[\alpha\gamma - \frac{\alpha^2}{2(1-\alpha)}, \frac{q-n}{q}(q - n - \gamma\sqrt{n}) - \frac{2\gamma^2}{(1-\gamma)(1-3\gamma)}]}.$$

When $q = n + \sqrt{n}$, the values $\gamma = 0.254$, $\alpha = 1 - \frac{1}{\sqrt{1+2\gamma}} = 0.186$ achieve the largest decrease $\delta = 0.026$ in the potential function and minimize the number of iterations.

From Theorems 1 and 2 it follows that Algorithm $\mathcal{A}$ finds an ϵ-optimal solution in $O(\sqrt{n}\ \log \frac{1}{\epsilon})$ iterations. In the next section we show that this bound is tight.

3 On the complexity of Algorithm $\mathcal{A}$ without line searches

We consider the following LP:

$$\begin{aligned} (P_1)\ \textit{minimize} \quad & \textstyle\sum_{i=1}^{\frac{n}{2}} x_i + 2^n \sum_{i=1}^{n} x_i \\ \textit{subject to} \quad & \textstyle\sum_{i=1}^{n} x_i = 1 \\ & x \geq 0, \end{aligned}$$

and its dual

$$\begin{aligned} (D_1)\ \textit{maximize} \quad & y \\ \textit{subject to} \quad & y + s_i = 2^n + 1, \quad i = 1, \ldots, \tfrac{n}{2} \\ & y + s_i = 2^n, \quad i = \tfrac{n}{2} + 1, \ldots, n \\ & s \geq 0, \end{aligned}$$

i.e., (P_1) is in standard form with $A = e_n$, $b = 1$, $c = f + 2^n e_n$, where

$$f = \begin{pmatrix} e_{\frac{n}{2}} \\ 0 \end{pmatrix}.$$

We use as initial point the point:

$$x^0 = \frac{1}{n} e_n, \quad s^0 = \frac{1}{2} e_n + f, \quad y^0 = -\frac{1}{2} + 2^n.$$

We first prove the following proposition:

Proposition 1 *Prior to the k-th iteration of Algorithm $\mathcal{A}$*

$$u_i^k = u_j^k,\ x_i^k = x_j^k,\ s_i^k = s_j^k, \quad i, j \leq \frac{n}{2}$$

$$u_i^k = u_j^k,\ x_i^k = x_j^k,\ s_i^k = s_j^k, \quad i, j > \frac{n}{2}.$$

Proof

By induction on k. For $k = 0$ the proposition is obviously true. Assuming that the proposition holds for k, we let $x_i^k = a_1, s_i^k = \theta_1$ for $i \leq \frac{n}{2}$ and $x_i^k = a_2, s_i^k = \theta_2$ for $i > \frac{n}{2}$. Then

$$AX^k = [a_1, \ldots, a_1, a_2, \ldots, a_2],$$

$$\bar{A}^k = (AX^k)'(AX^kX^kA')^{-1}AX^k = \frac{1}{\frac{n}{2}(a_1^2 + a_2^2)} \begin{pmatrix} a_1^2 U_{\frac{n}{2}} & a_1 a_2 U_{\frac{n}{2}} \\ a_1 a_2 U_{\frac{n}{2}} & a_2^2 U_{\frac{n}{2}} \end{pmatrix},$$

where $U_l = e_l e_l'$ is an $l \times l$ matrix, in which all entries are 1.

Since

$$\frac{q}{(x^k)'s^k} X^k s^k - e_n = \begin{pmatrix} (\frac{qa_1\theta_1}{\frac{n}{2}(a_1\theta_1 + a_2\theta_2)} - 1)e_{\frac{n}{2}} \\ (\frac{qa_2\theta_2}{\frac{n}{2}(a_1\theta_1 + a_2\theta_2)} - 1)e_{\frac{n}{2}} \end{pmatrix} = \begin{pmatrix} p_1 e_{\frac{n}{2}} \\ p_2 e_{\frac{n}{2}} \end{pmatrix},$$

we obtain

$$u^k = \begin{pmatrix} (p_1 - \frac{a_1^2 p_1 + a_1 a_2 p_2}{a_1^2 + a_2^2})e_{\frac{n}{2}} \\ (p_2 - \frac{a_2^2 p_2 + a_1 a_2 p_1}{a_1^2 + a_2^2})e_{\frac{n}{2}} \end{pmatrix}.$$

If $||u^k|| \geq \gamma$, $x^{k+1} = x^k - \alpha \frac{X^k u^k}{||u^k||}$, $s^{k+1} = s^k$, and therefore the proposition follows by induction. If $||u^k|| < \gamma$, $x^{k+1} = x^k$, $s^{k+1} = \frac{(x^k)'s^k}{q}(X^k)^{-1}(u^k + e_n)$ and the proposition follows by induction also. □

We next show that at every iteration, the gap between the primal objective value and the optimal value does not decrease by a factor of more than $1 - O(\frac{1}{\sqrt{n}})$.

Proposition 2 *The primal objective values before and after iteration k of Algorithm $\mathcal{A}$ satisfy:*

$$\frac{c'(x^{k+1} - x^*)}{c'(x^k - x^*)} \geq (1 - \frac{\sqrt{2}\alpha}{\sqrt{n}}).$$

where x^* is an optimal solution to (P_1).

Proof

Prior to iteration k, $c'(x^k - x^*) = \sum_{i=1}^{\frac{n}{2}} x_i^k = \frac{n}{2} x_1^k$, from Proposition 1. Therefore, in order to prove the proposition it suffices to show that $x_1^{k+1} \geq x_1^k(1 - \frac{\sqrt{2}\alpha}{\sqrt{n}})$.

If a dual step is executed at the k–th iteration $x_1^{k+1} = x_1^k$, and the proposition trivially follows.

If a primal is executed at the k–th iteration

$$x_1^{k+1} = (1 - \alpha \frac{u_1^k}{||u^k||})\, x_1^k.$$

From Proposition 1, $u_1^k = \ldots = u_{\frac{n}{2}}^k$, and therefore

$$\frac{|u_1^k|}{||u^k||} \leq \frac{|u_1^k|}{\sqrt{\frac{n}{2}}|u_1^k|} = \frac{\sqrt{2}}{\sqrt{n}}.$$

Therefore,

$$x_1^{k+1} = (1 - \alpha \frac{u_1^k}{||u^k||})\, x_1^k \geq (1 - \frac{\sqrt{2}\alpha}{\sqrt{n}})\, x_1^k,$$

which completes the proof of the proposition. □

We are now ready to prove the main result of this section.

Theorem 3 *For $n \geq \frac{32}{9}\alpha^2$ Algorithm $\mathcal{A}$ needs at least*

$$\frac{\sqrt{n}}{2\sqrt{2}\alpha} \log \frac{1}{2\epsilon}$$

to find an ϵ-optimal solution to the linear program (P_1), if it starts from $x^0 = \frac{1}{n}e_n$, $s^0 = \frac{1}{2}e_n + f$, $y^0 = -\frac{1}{2} + 2^n$, where f is defined the same as before.

Proof

Suppose the algorithm found an ϵ-optimal solution at step k^*. Then

$$(x^{k^*})' s^{k^*} = c' x^{k^*} - y^{k^*} \leq \epsilon.$$

Algorithm $\mathcal{A}$ generates at each step dual feasible solutions (s^k, y^k). In (D_1) $s_i^k + y^k = 2^n$ for $i > \frac{n}{2}$ with $s_i \geq 0$, which implies that $y^k \leq 2^n$. Then

$$c' x^{k^*} \leq \epsilon + y^{k^*} \leq \epsilon + 2^n.$$

From Proposition 2, since $c' x^* = 2^n$,

$$c' x^{k^*} - 2^n \geq (1 - \frac{\sqrt{2}\alpha}{\sqrt{n}})^{k^*} (c' x^0 - 2^n).$$

Therefore, since $c' x^0 = \frac{1}{2} + 2^n$ we obtain

$$(1 - \frac{\sqrt{2}\alpha}{\sqrt{n}})^{k^*} \leq 2\epsilon,$$

which implies that

$$k^* \log (1 - \frac{\sqrt{2}\alpha}{\sqrt{n}}) \leq \log (2\epsilon).$$

But $\log (1 - x) > -2x$ for $0 \leq x \leq \frac{3}{4}$. Therefore, for $\frac{\sqrt{2}\alpha}{\sqrt{n}} \leq \frac{3}{4}$, i.e., $n \geq \frac{32}{9}\alpha^2$,

$$k^* \geq \frac{\sqrt{n}}{2\sqrt{2}\alpha} \log \frac{1}{2\epsilon}. \quad \square$$

The previous example shows that as long as we use a fixed α (no line searches), algorithm $\mathcal{A}$ needs at least $\Omega(\sqrt{n}\log\frac{1}{\epsilon})$ iterations to find an ϵ-optimal solution for all values of $q \geq n$ and in particular for $q = n+\sqrt{n}$. Since for $q = n+\sqrt{n}$ algorithm $\mathcal{A}$ takes at most $O(\sqrt{n}\log\frac{1}{\epsilon})$ iterations to find an ϵ-optimal solution, we establish that in this case the bound is tight. We note that in their analysis Ye [12], Freund [3], and Gonzaga and Todd [5] use a fixed α and do not consider the impact of line searches on the complexity.

If we want to translate our results to the usual complexity measure L, which measures input the size of a linear program (P), we remark that $L = \Omega(n)$ in example (P_1). The initial potential function

$$G(x^0, s^0) = n\log n - \frac{n}{2}\log\frac{3}{4} = O(\sqrt{n}L).$$

Since $G(x^0, s^0) = O(\sqrt{n}L)$, by selecting $\epsilon = 2^{-2L}$, and then using Theorems 2 and 3, we obtain

Corollary 1 *Algorithm $\mathcal{A}$ takes $\Theta(\sqrt{n}L)$ iterations to find an exact solution for problem (P_1).*

We remark that although we only showed that the bound $O(\sqrt{n}L)$ is tight for Algorithm $\mathcal{A}$, the analysis easily extends to the algorithms of Freund [3] and Gonzaga and Todd [5]. Proposition 1 can also be proven inductively and, since both these algorithms have exactly the same primal steps as Algorithm $\mathcal{A}$, Proposition 2 also holds. As a result, the bound $O(\sqrt{n}L)$ is tight for these algorithms as well.

4 On the complexity of Algorithm $\mathcal{A}$ with line searches

The analysis in the previous section did not address the case in which we allow line searches in the potential function, i.e., we choose α in each step to achieve the most decrease in the potential function. One might hope that line searches are not only practically useful, but could also be used to improve the complexity of Algorithm $\mathcal{A}$.

We show in this section that line searches do not improve the complexity as long as $q \leq n + \sqrt{n}$.

We first prove some properties of Algorithm $\mathcal{A}$ that hold for an arbitrary linear program. Let $z^k = (x^k)'s^k$ be the duality gap after the execution of the $k-1$st step of the algorithm.

Proposition 3 *If $q \leq n+\sqrt{n}$, then after a dual step at step $k-1$, the duality gap satisfies:*

$$\frac{z^k}{z^{k-1}} \geq 1 - \frac{2}{\sqrt{n}}.$$

Proof

After a dual step at step $k-1$ of the algorithm

$$\begin{aligned}
z^k &= (s^k)'x^k \\
&= \frac{(x^{k-1})'s^{k-1}}{q}[(X^{k-1})^{-1}(u^{k-1}+e_n)]'x^{k-1} \\
&= \frac{z^{k-1}}{q}e_n'(u^{k-1}+e_n) \\
&\geq z^{k-1}\frac{n-||u^{k-1}||_1}{q} \\
&\geq z^{k-1}\frac{n-\sqrt{n}||u^{k-1}||_2}{q} \\
&\geq z^{k-1}\frac{n-\gamma\sqrt{n}}{n+\sqrt{n}} \\
&\geq z^{k-1}\frac{n-\sqrt{n}}{n+\sqrt{n}} \\
&\geq z^{k-1}(1-\frac{2}{\sqrt{n}}),
\end{aligned}$$

which completes the proof of the proposition. □

Proposition 4 *After a dual step at step $k-1$,*

$$\frac{x_i^k s_i^k}{x_j^k s_j^k} \leq \frac{1+\gamma}{1-\gamma}, \text{ for all } i,j.$$

Proof

After a dual step at step $k-1$,

$$X^k s^k = \frac{(x^{k-1})'s^{k-1}}{q}(u^{k-1}+e_n).$$

Since $|u_i^{k-1}| \leq ||u^{k-1}||_2 < \gamma$, we obtain the proposition. □

In order to prove that line searches do not improve the complexity we consider the following LP:

$$\begin{aligned}
(P_2)\ \textit{minimize} \quad & \textstyle\sum_{i=1}^n x_i \\
\textit{subject to} \quad & x_i + \delta x_n = 1,\ i = 1,\ldots,n-1 \\
& x \geq 0,
\end{aligned}$$

and its dual

$$\begin{aligned} (D_2)\ \textit{maximize} \quad & \textstyle\sum_{i=1}^{n-1} y_i \\ \textit{subject to} \quad & y_i + s_i = 1, \quad i = 1, \ldots, n-1 \\ & \delta \textstyle\sum_{i=1}^{n-1} y_i + s_n = 1 \\ & s \geq 0, \end{aligned}$$

i.e., problem (P_2) is in standard form with $A = (I_{n-1}, \delta e_{n-1})$, $b = e_{n-1}$, $c = e_n$, where $0 < \delta < \frac{1}{n-1}$.

We use as initial point the point:

$$x^0 = \frac{1}{1+\delta} e_n, \quad s^0 = e_n, \quad y^0 = 0.$$

Note that the optimal solution is

$$x^* = \begin{pmatrix} e_{n-1} \\ 0 \end{pmatrix}, \quad s^* = \begin{pmatrix} 0 \\ 1 - (n-1)\delta \end{pmatrix}.$$

Let us observe that at every step of the algorithm $x_i^k = 1 - \delta x_n^k$ for all $i = 1, \ldots, n-1$.

The strategy for showing the lower bound on the number of iterations even if we permit line searches is as follows:

1. We first establish that between successive dual steps the duality does not decrease by a factor of more than $1 - O(\frac{1}{n})$.

2. We then use Proposition 3 to show that after each dual step the duality gap does not decrease by a factor of more than $1 - O(\frac{1}{\sqrt{n}})$.

3. Combining these observations we prove the main theorem of this section.

Suppose Algorithm $\mathcal{A}$ stops at step k^*. The algorithm starts with the initial point (x^0, s^0), takes a number (possibly zero) of primal steps, then takes a dual step at step k_1, then a number (possibly zero) of primal steps, a dual step at step k_2 and so on. The dual steps are taken at steps k_r, $r = 1, \ldots, m$.

We first consider the decrease of the duality gap between consecutive dual steps. Let k_r, k_{r+1}, $(r = 1, \ldots, m)$ be two consecutive dual steps in the execution of the algorithm, i.e., all steps between these two steps are primal steps.

Proposition 5 *a) For $r = 1, \ldots, m$ the duality gap just before k_{r+1} satisfies:*

$$z^{k_{r+1}} \geq (1 - \frac{1+\gamma}{(1-\gamma)n}) z^{k_r+1}.$$

b) Just prior to the first dual step we have

$$z^{k_1} \geq (1 - \frac{1+\gamma}{(1-\gamma)n}) z^0.$$

c) The duality gap at last step satisfies

$$z^{k^*} \geq (1 - \frac{1+\gamma}{(1-\gamma)n}) z^{k_m+1}.$$

Proof

Let k be a primal step between k_r and k_{r+1}. The result holds trivially if no such k exists. Then prior to the k–th step

$$\begin{aligned}
z^k &= (x^k)' s^k \\
&= \sum_{i=1}^{n-1} x_i^k (1 - y_i^k) + x_n^k (1 - \delta \sum_{i=1}^{n-1} y_i^k) \\
&= (1 - \delta x_n^k) \sum_{i=1}^{n-1} (1 - y_i^k) + x_n^k (1 - \delta \sum_{i=1}^{n-1} y_i^k) \\
&= \sum_{i=1}^{n-1} (1 - y_i^k) + (1 - (n-1)\delta) x_n^k \\
&= \sum_{i=1}^{n-1} s_i^k + (1 - (n-1)\delta) x_n^k \\
&= \sum_{i=1}^{n-1} s_i^{k_r+1} + (1 - (n-1)\delta) x_n^k,
\end{aligned}$$

since during all primal steps the dual slacks s^k remain unchanged. Since $\delta < \frac{1}{n-1}$, we obtain that

$$z^k \geq \sum_{i=1}^{n-1} s_i^{k_r+1} \geq (1 - \delta x_n^{k_r+1}) \sum_{i=1}^{n-1} s_i^{k_r+1} = \sum_{i=1}^{n-1} x_i^{k_r+1} s_i^{k_r+1} = z^{k_r+1} - x_n^{k_r+1} s_n^{k_r+1}.$$

Applying Proposition 4 after the execution of the dual step k_r we will have

$$x_n^{k_r+1} s_n^{k_r+1} \leq \frac{1}{n} \frac{1+\gamma}{1-\gamma} \sum_{i=1}^{n} x_i^{k_r+1} s_i^{k_r+1}.$$

Therefore,

$$z^k \geq (1 - \frac{1+\gamma}{(1-\gamma)n}) z^{k_r+1}.$$

Applying the above inequality for $k = k_{r+1}$ just prior to the dual step k_{r+1}, we prove part (a) of the proposition. Applying the inequality for $(k = k_1)$ and $(k = k^*)$ we prove parts (b) and (c) of the proposition respectively. □

We are now ready to prove the main theorem of this section.

Theorem 4 *For $q \leq n + \sqrt{n}$, then for $n \geq \max[8, \frac{4}{3}\frac{1+\gamma}{1-\gamma}, (\frac{1+\gamma}{2(1-\gamma)})^2]$, Algorithm $\mathcal{A}$ needs at least*

$$\frac{\sqrt{n}}{8}(\log n + \log \frac{1}{2\epsilon})$$

iterations to find an ϵ-optimal solution to the linear program (P_2), when it starts from $x^0 = \frac{1}{1+\delta}e_n$, $s^0 = e_n$, $y^0 = 0$ even if the algorithm performs line searches in the potential function.

Proof

Suppose the algorithm found an ϵ-optimal solution at step k^*. As mentioned earlier the algorithm starts with the initial point, it takes a number (possibly zero) of primal steps, then takes a dual step at step k_1, then a number (possibly zero) of primal steps, a dual step at step k_2 and so on. The dual steps are taken at steps k_r, $r = 1, \ldots, m$.

From Proposition 5(b) we have that prior to the first dual step

$$z^{k_1} \geq (1 - \frac{1+\gamma}{(1-\gamma)n})z^0 = (1 - \frac{1+\gamma}{(1-\gamma)n})\frac{n}{1+\delta}.$$

For $q \leq n + \sqrt{n}$, we obtain from Proposition 3 that for $r = 1, \ldots, m$

$$z^{k_r+1} \geq (1 - \frac{2}{\sqrt{n}})z^{k_r}.$$

From Proposition 5(a) we obtain that for $r = 1, \ldots, m$

$$z^{k_r+1} \geq (1 - \frac{1+\gamma}{(1-\gamma)n})z^{k_r+1}.$$

From Proposition 5(c) we obtain

$$z^{k^*} \geq (1 - \frac{1+\gamma}{(1-\gamma)n})z^{k_m+1}.$$

Since the algorithm stopped at step k^*, $z^{k^*} \leq \epsilon$. Combining these inequalities we obtain

$$(1 - \frac{2}{\sqrt{n}})^{m+1}(1 - \frac{1+\gamma}{(1-\gamma)n})^{m+1}\frac{n}{1+\delta} \leq \epsilon,$$

Since $\log(1-x) > -2x$ for $0 \leq x \leq \frac{3}{4}$, taking logarithms in the last inequality and using $\delta < \frac{1}{n-1} < 1$ we obtain that for $n \geq \max[8, \frac{4}{3}\frac{1+\gamma}{1-\gamma}]$

$$m+1 \geq \frac{\sqrt{n}}{4(1+\frac{1+\gamma}{2(1-\gamma)\sqrt{n}})} \log \frac{n}{2\epsilon} \geq \frac{\sqrt{n}}{8}(\log n + \log \frac{1}{2\epsilon}),$$

where the last inequality follows for $n \geq (\frac{1+\gamma}{2(1-\gamma)})^2$.

Since $k^* \geq m+1$ the theorem follows. □

The proof of Theorem 4 reveals that the fundamental reason that line searches do not help in the complexity is that we need to take a large number of dual steps to decrease the duality gap. On the other hand, because of the special structure of the problem the primal steps do not decrease the duality gap substantially, even if line searches are executed. By choosing δ to be small enough, L can be set to be arbitrarily large, so that the starting point in Theorem 4 satisfies $G(x^0, s^0) = O(\sqrt{n}L)$. Theorems 2 and 4 then imply that Algorithm $\mathcal{A}$ takes $\Theta(\sqrt{n}L)$ iterations to find an exact solution for problem (P_2).

5 Reflections

One might ask what is the *deep reason* that $\sqrt{n}$ appears in the number of iterations, i.e., why $\sqrt{n}$ and not some other function of n?

The reason that $\sqrt{n}$ appears in the complexity for the case without line searches is in Propositions 2 and 3. The algorithm uses the test $||u^k||_2 \geq \gamma$ to control whether a primal or dual step is taken. The decrease in the potential function when a primal step is executed is measured through $||u^k||_2$. On the other hand, after a dual step the dual slacks are updated proportionally to $(X^k)^{-1}(u^k + e_n)$. The decrease in the potential function after a dual step is executed is affected by $||u^k||_1$, not $||u^k||_2$. The only information we have regarding $||u^k||_1$ is that

$$||u^k||_1 \leq \sqrt{n}||u^k||_2,$$

and this inequality is tight. It is this $\sqrt{n}$, which affects the number of iterations. It appears then, that the key reason, at least at a superficial level, for the $\sqrt{n}$ is the use of different norms in the primal and dual steps.

At a somewhat deeper level, we conjecture that it is the use of the logarithmic potential function either in the construction of an algorithm or in its proof that leads to $\sqrt{n}$, i.e., we conjecture

Conjecture 1 *An interior point algorithms that uses the logarithmic barrier function either in its construction or in its proof needs at least $\Omega(\sqrt{n}L)$ iterations to find an optimal solution in the worst case.*

All known proofs of polynomial number of iterations for interior point algorithms have some relation with the logarithmic barrier function. Algorithm $\mathcal{A}$ uses the logarithmic potential function to design the algorithm directly, while many other algorithms use logarithmic potential functions to bound the number of iterations. If the conjecture is correct, it would imply that all known polynomial interior point algorithms need at least $\Omega(\sqrt{n}L)$ iterations in the worst case.

Another interesting observation is that the reasons we achieved the $\Omega(\sqrt{n}\log\frac{1}{\epsilon})$ bound on the number of iterations are quite different in the two examples.

In the example of Section 3, the key reason was Proposition 2 that the gap between the primal objective value (which is not affected by dual steps) and the optimal value can not be decreased by a factor of more than $1 - O(\frac{1}{\sqrt{n}})$ after a primal step. So, in the first example, the primal steps were central to the lower bound.

In the example of Section 4, the key reason was Proposition 3 that after a dual step the duality gap does not decrease by a factor of more than $1 - O(\frac{1}{\sqrt{n}})$. Note that this result holds for an arbitrary linear program, not only for the particular example. The particular structure of the example was used to show Proposition 5 that the primal steps between duals do not decrease the duality gap by a factor of more than $1 - O(\frac{1}{n})$. In this case both the primal and dual steps were central to the lower bound. We also remark that while the first example works for Ye [12], Freund [3], and Gonzaga and Todd [5], the second example only works for Ye's algorithm.

We have left open the case of line searches with $q > n + \sqrt{n}$. We conjecture that also in this case an example can be found to achieve the $\Theta(\sqrt{n}\log\frac{1}{\epsilon})$ bound.

Much of the work in the complexity of interior point algorithms has focused on the worst case, which is not in agreement with the practical experience. The observed behavior of interior point algorithms is $O(\log n\, L)$ rather than $O(\sqrt{n}\, L)$ iterations. Although some research on anticipative behavior has been completed, a genuine and rigorous understanding of the average behavior of interior point algorithms is still missing.

We feel that the understanding of Conjecture 1 and of the average behavior of interior point algorithms are probably the most interesting directions of research in the theory of

interior point algorithms.

Acknowledgements

Both authors would like to thank Professor Dimitri Bertsekas for his encouragement and for several interesting discussions on the field of interior point algorithms. The first author would also like to thank his colleague Professor Robert Freund for stimulating his interest in interior point algorithms over the years.

References

[1] K. M. Anstreicher, (1989), "The worst-case Step in Karmarkar's algorithm", *Mathematics of Operations Research*, **14**, 295-302.

[2] K. M. Anstreicher, (1991), "On the performance of Karmarkar's algorithm over a sequence of iterations", *SIAM J. Optimization*, **1**, 22-29.

[3] R. Freund, (1991), "Polynomial-time algorithms for linear programming based only on primal scaling and projected gradients of a potential function", *Mathematical Programming*, **51**, 203-222.

[4] C. C. Gonzaga, (1988), "An algorithm for solving linear programming problems in $O(n^3L)$ operations", in N. Megiddo, ed., *Progress in Mathematical Programming, Interior Point and Related Methods*, Springer, New York, 1-28.

[5] C. C. Gonzaga and M. J. Todd, (1992), "An $O(\sqrt{n}L)$-iteration large-step primal-dual affine algorithm for linear programming", *SIAM J. Optimization*, **2**, 349-359.

[6] J. Kaliski and Y. Ye, (1991), "Convergence behavior of Karmarkar's projective algorithm for solving a simple linear program", *Operations Research Letters*, **10**, 389-393.

[7] N. Karmarkar, (1984), "A new polynomial-time algorithm for linear programming", *Combinatorica*, **4**, 373-395.

[8] C. McDiarmid, (1990), "On the improvement per iteration in Karmarkar's algorithm for linear programming", *Mathematical Programming*, **46**, 299-320.

[9] C. Papadimitriou and K. Steiglitz, (1982), *Combinatorial Optimization; Algorithms and Complexity*, Prentice Hall.

[10] J. Renegar, (1988), "A polynomial-time algorithm, based on Newton's method, for linear programming", *Mathematical Programming*, **40**, 59-93.

[11] M. J. Todd and Y. Ye, (1990), "A centered projective algorithm for linear programming", *Mathematics of Operations Research*, **15**, 508-529.

[12] Y. Ye, (1991), "An $O(n^3L)$ potential reduction algorithm for linear programming", *Mathematical Programming*, **50**, 239-258.