A Simplified Homogeneous and Self-Dual Linear Programming Algorithm and Its Implementation*

Xiaojie Xu † Pi-Fang Hung ‡ Yinyu Ye §

September 1993 (revised November 1994)

Abstract: We present a simplification and generalization of the recent homogeneous and self-dual linear programming (LP) algorithm. The algorithm does not use any Big-M initial point and achieves $O(\sqrt{n}L)$-iteration complexity, where n and L are the number of variables and the length of data of the LP problem. It also detects LP infeasibility based on a provable criterion. Its preliminary implementation with a simple predictor and corrector technique results in an efficient computer code in practice. In contrast to other interior-point methods, our code solves NETLIB problems, feasible or infeasible, starting simply from $x = e$ (primal variables), $y = 0$ (dual variables), $z = e$ (dual slack variables), where e is the vector of all ones. We describe our computational experience in solving these problems, and compare our results with OB1.60, a state of the art implementation of interior-point algorithms.

Key words: Linear Programming, homogeneous and self-dual linear feasibility model, predictor-corrector algorithm, implementation.

*Research supported in part by NSF Grant DDM-9207347 and by the Iowa College of Business Administration Summer Grant.

†Institute of Systems Science, Academia Sinica, Beijing 100080, China. The author is currently visiting Department of Management Sciences, The University of Iowa, Iowa City, Iowa 52242, USA.

‡Department of Mathematics, The University of Iowa, Iowa City, Iowa 52242, USA. Current address: Department of Mathematics, The Tunghai University, No 181, Section 3, Taichung-kan Road, Taichung, Taiwan 40704.

§Department of Management Sciences, The University of Iowa, Iowa City, Iowa 52242, USA. Part of this work was done while the author was on a sabbatical leave from the University of Iowa and visiting the Cornell Theory Center, Cornell University, Ithaca, NY 14853, USA, supported in part by the Cornell Center for Applied Mathematics and by the Advanced Computing Research Institute, a unit of the Cornell Theory Center, which receives major funding from the National Science Foundation and IBM Corporation, with additional support from New York State and members of its Corporate Research Institute.

1 Introduction

Consider the linear programming (LP) problem in the standard form:

$$\begin{array}{lll} (LP) & \text{minimize} & c^T x \\ & \text{subject to} & Ax = b, \ x \geq 0, \end{array}$$

where $c \in R^n$, $A \in R^{m \times n}$ and $b \in R^m$ are given, $x \in R^n$, and ^{T} denotes transpose. (LP) is said to be feasible if and only if its constraints are consistent; it is called unbounded if there is a sequence $\{x^k\}$ such that x^k is feasible for all k but $c^T x^k \to -\infty$. (LP) has a solution if and only if it is feasible and bounded.

The dual problem of (LP) can be written as

$$\begin{array}{lll} (LD) & \text{maximize} & b^T y \\ & \text{subject to} & A^T y \leq c, \end{array}$$

where $y \in R^m$. We call $z = c - A^T y \in R^n$ dual slacks.

Recently Ye-Todd-Mizuno [15] developed a homogeneous and self-dual (HLP) linear programming algorithm based on the construction of a homogeneous and self-dual (artificial) LP model, in which the dimension of the problem is increased by 2. In this paper we construct a homogeneous and self-dual linear feasibility model (HLF) with its dimension increased by 1. (Our model essentially ignores the normalizing constraint of Ye-Todd-Mizuno's model.) Applying a modified Newton method, we develop an $O(\sqrt{n}L)$ algorithm for solving feasible or infeasible LP problems. It can be regarded as a simplification, as well as a generalization, of Ye-Todd-Mizuno's algorithm. Their algorithm decreases duality gap and infeasibility error at the same rate, while ours allows different reduction rates in some iterations. Similarly, the algorithm possesses the following features:

- It achieves $O(\sqrt{n}L)$-iteration complexity, the best known theoretical result up to date.
- It solves LP problems without any regularity assumption concerning the existence of optimal, feasible, or interior feasible solutions.
- It can start at any positive primal-dual pair, feasible or infeasible, near the central ray of the positive orthant (cone), and it does not use any Big M penalty parameter or lower bound.
- Each iteration factorizes the same matrix as in standard (primal-dual) interior-point algorithms, but with one more forward and backward substitution solve.
- If the LP problem has a solution, the algorithm generates a sequence that approaches feasibility and optimality simultaneously; if the problem is infeasible or unbounded, the algorithm correctly detects infeasibility for at least one of the primal and dual problems.

Using Mehrotra's simple predictor and corrector strategy (Mehrotra [8]), i.e, taking both predictor and corrector steps with the same matrix factors in one iteration, we develop an efficient implementation code in practice. We propose a technique allowing different step sizes in primal and dual updates. We report our computational results in solving the NETLIB feasible problems, with a comparison to OB1.60, a state of the art implementation of interior-point algorithms. We run both OB1 and our code on an APOLLO 720 workstation with compiler F77 +O3.

Very recently, a set of infeasible LP problems has joined the NETLIB collection. Some of them are quite interesting such as CPLEX2, an almost-feasible problem contributed by CPLEX Optimization Inc. Thus, in addition to solving the NETLIB feasible problems, we also test our code in solving these infeasible problems.

Starting simply from $x = e$, $y = 0$, $z = e$ (dual slack) and using the default values for program parameters, our code successfully solves or detects infeasibility of all encountered test problems in NETLIB. As we understand, it is a major problematical issue to detect infeasibility for practical interior-point methods. Some theoretical results hold only for feasible cases. Other approaches to detect infeasibility are somewhat difficult to implement in practice (Lustig et al. [6]). Thus, our method may be a reliable alternative to resolve this issue.

2 A homogeneous and self-dual linear feasibility model

The homogeneous and self-dual linear feasibility model has the form

$$\begin{array}{lrrrl} (HLF) & & Ax & -b\tau & = 0, \\ & -A^T y & & +c\tau & \geq 0, \\ & b^T y & -c^T x & & \geq 0, \\ & y \text{ free}, & x \geq 0, & \tau \geq 0. & \end{array} \tag{1}$$

This system was first proposed and studied by Goldman and Tucker [2][12].

Denote by z the slack vector for the second (inequality) constraint and by κ the slack scalar for the third (inequality) constraint. Then, we are interested in finding a strictly complementary point such that

$$x^T z = 0 \quad \text{and} \quad \tau\kappa = 0.$$

We have the following theorem for (HLF), which is similar to Theorem 1 of [15] and is based on Goldman and Tucker [2][12].

Theorem 1 .

i. *Consider (HLF) as an LP problem with a zero right-hand and a zero objective vector. Then, (HLF) is self-dual, i.e., the dual of (HLF) has the identical form as (HLF).*

ii. *(HLF) is feasible and it has no interior or strictly feasible point* $(y, x > 0, \tau > 0, s > 0, \kappa > 0)$.

iii. *There is a complementary solution or ray* $(y^*, x^*, \tau^*, z^*, \kappa^*)$ *such that*

$$\begin{pmatrix} x^* + z^* \\ \tau^* + \kappa^* \end{pmatrix} > 0,$$

which we called a strictly complementary solution or ray.

Let $(y^*, x^*, \tau^*, z^*, \kappa^*)$ *be a strictly complementary solution for (HLF). Then*

iv. *(LP) has a solution if and only if* $\tau^* > 0$. *In this case,* x^*/τ^* *is an optimal solution for (LP) and* $(y^*/\tau^*, z^*/\tau^*)$ *is an optimal solution for (LD).*

v. *If* $\tau^* = 0$, *then* $\kappa^* > 0$, *which implies that* $c^T x^* - b^T y^* < 0$, *i.e., at least one of* $c^T x^*$ *and* $-b^T y^*$ *is strictly less than zero. If* $c^T x^* < 0$ *then (LD) is infeasible; if* $-b^T y^* < 0$ *then (LP) is infeasible; and if both* $c^T x^* < 0$ *and* $-b^T y^* < 0$ *then both (LP) and (LD) are infeasible.*

3 The $O(\sqrt{n}L)$-iteration algorithm

Let $(y^k,\ x^k > 0, \tau^k > 0, z^k > 0, \kappa^k > 0)$ be an (infeasible) interior point for (HLF), and define

$$r_P^k = b\tau^k - Ax^k, \quad r_D^k = c\tau^k - A^T y^k - z^k, \quad r_G^k = c^T x^k - b^T y^k + \kappa^k. \tag{2}$$

Then, starting from this interior point, we apply a modified Newton method directly to solving (HLF). In each iteration, the method basically solves the following system of linear equations for direction $(d_y, d_x, d_\tau, d_z, d_\kappa)$, as proposed by Kojima, Megiddo and Mizuno [3],

$$\begin{array}{rrrrrcl} & A\ d_x & -b\ d_\tau & & & = & \eta\ r_P^k, \\ -A^T\ d_y & & +c\ d_\tau & -d_z & & = & -\eta\ r_D^k, \\ b^T\ d_y & -c^T\ d_x & & & -d_\kappa & = & \eta\ r_G^k, \end{array} \tag{3}$$

and

$$\begin{array}{rcl} X^k d_z + Z^k d_x & = & \gamma\mu^k e - X^k z^k, \\ \tau^k d_\kappa + \kappa^k d_\tau & = & \gamma\mu^k - \tau^k \kappa^k, \end{array} \tag{4}$$

where

$$\mu^k = ((x^k)^T z^k + \tau^k \kappa^k)/(n+1),$$

and η, γ are positive scalar parameters. We have the following key lemma.

Lemma 2 . *The direction defined by (3) and (4) satisfies:*

$$(d_x)^T d_z + d_\tau d_\kappa = \eta(1 - \gamma - \eta)(n+1)\mu^k.$$

Proof. From the skew-symmetric property of the system

$$\begin{aligned}(d_x)^T d_z + d_\tau d_\kappa &= (d_x)^T[-A^T d_y + c d_\tau + r_D^k \eta] + d_\tau[b^T d_y - c^T d_x - r_G^k \eta] \\ &= -(A d_x)^T d_y + (d_x)^T r_D^k \eta + b^T d_y d_\tau - r_G^k d_\tau \eta \\ &= -[(d_y)^T r_P^k - (d_x)^T r_D^k + d_\tau r_G^k]\ \eta\end{aligned} \tag{5}$$

Since any (y, x, z, τ, κ) and the corresponding (r_P, r_D, r_G) defined by (2) satisfy

$$y^T r_P - x^T r_D + \tau r_G = x^T z + \tau\kappa,$$

we have, for both $(y^k, x^k, z^k, \tau^k, \kappa^k)$ and $(y^k + d_y, x^k + d_x, z^k + d_z, \tau^k + d_\tau, \kappa^k + d_\kappa)$,

$$(y^k)^T r_P^k - (x^k)^T r_D^k + \tau^k r_G^k = (x^k)^T z^k + \tau^k \kappa^k \tag{6}$$

and

$$[(y^k + d_y)^T r_P^k - (x^k + d_x)^T r_D^k + (\tau^k + d_\tau) r_G^k](1-\eta) = (x^k + d_x)^T (z^k + d_z) + (\tau^k + d_\tau)(\kappa^k + d_\kappa).$$

Hence

$$\begin{aligned}&[(x^k)^T z^k + \tau^k\kappa^k](1-\eta) + [(d_y)^T r_P^k - (d_x)^T r_D^k + (d_\tau) r_G^k](1-\eta) \\ = \ &[(x^k)^T z^k + \tau^k\kappa^k] + [(d_x)^T d_z + d_\tau d_\kappa] + [(x^k)^T d_z + (d_x)^T z^k + \tau^k d_\kappa + d_\tau \kappa^k].\end{aligned}$$

This relation together with (5) imply that

$$\begin{aligned}(d_y)^T r_P^k - (d_x)^T r_D^k + (d_\tau) r_G^k &= [(x^k)^T z^k + \tau^k\kappa^k]\eta + [(x^k)^T d_z + (d_x)^T z^k + \tau^k d_\kappa + d_\tau\kappa^k] \\ &= \eta(n+1)\mu^k + [(x^k)^T d_z + (d_x)^T z^k + \tau^k d_\kappa + d_\tau \kappa^k].\end{aligned} \tag{7}$$

Finally, from (4) we have

$$\begin{aligned}(x^k)^T d_z + (z^k)^T d_x + \tau^k d_\kappa + \kappa^k d_\tau &= \gamma\mu^k(n+1) - (x^k)^T z^k - \tau^k \kappa^k \\ &= (\gamma - 1)(n+1)\mu^k,\end{aligned} \tag{8}$$

and combining (5), (7) and (8) gives

$$(d_x)^T d_z + d_\tau d_\kappa = -\eta[(d_y)^T r_P^k - (d_x)^T r_D^k + (d_\tau) r_G^k] = \eta(1-\gamma-\eta)(n+1)\mu^k.$$

□

Depending on the choice of parameters η and γ, the inner product of (d_x, d_τ) and (d_z, d_κ) has different signs.

- if $\eta = 1-\gamma$, then $(d_x)^T d_z + d_\tau d_\kappa = 0$;
- if $\eta < 1-\gamma$, then $(d_x)^T d_z + d_\tau d_\kappa > 0$;
- if $\eta > 1-\gamma$, then $(d_x)^T d_z + d_\tau d_\kappa < 0$.

The first case is like feasible LP algorithms—the primal and dual directions are orthogonal, the second case is like feasible strictly convex quadratic programming (QP) algorithms—the two directions make an acute angle, and the third case is like feasible concave QP algorithms. Thus, if we choose the parameters as in the first two cases, we should have the same polynomial convergence property as that in feasible LP and convex QP algorithms. However, we would like to mention that if we choose the parameters as in the third case and control

$$|(d_x)^T d_z + d_\tau d_\kappa| \le O(\mu^k), \tag{9}$$

we can still achieve a polynomial convergence result. We now describe a generic homogeneous and self-dual algorithm.

Generic HLF algorithm

Given initial point y^0, $x^0 > 0, \tau^0 > 0, z^0 > 0, \kappa^0 > 0$, $k \leftarrow 0$.

While (stopping criteria not satisfied) do

1. Let $r_P^k = b\tau^k - Ax^k$, $\quad r_D^k = c\tau^k - A^T y^k - z^k$, $\quad r_G^k = c^T x^k - b^T y^k + \kappa^k$.
2. Solve (3) and (4) for $(d_y, d_x, d_\tau, d_z, d_\kappa)$.
3. Choose a step size $\alpha^k > 0$.
4. Update

$$\begin{aligned} x^{k+1} &= x^k + \alpha^k d_x > 0, \\ y^{k+1} &= y^k + \alpha^k d_y, \\ z^{k+1} &= z^k + \alpha^k d_z > 0, \\ \tau^{k+1} &= \tau^k + \alpha^k d_\tau > 0, \\ \kappa^{k+1} &= \kappa^k + \alpha^k d_\kappa > 0. \end{aligned} \tag{10}$$

5. $k \leftarrow k + 1$.

Lemma 3 . *The generic algorithm generates $\{\mu^k\}$ and $\{\theta^k\}$ satisfying*

$$\mu^0 = [(x^0)^T z^0 + \tau^0 \kappa^0] \ / \ (n+1), \quad \mu^{k+1} = (1 - \alpha^k \eta)[1 - \alpha^k(1 - \gamma - \eta)] \ \mu^k,$$

and

$$\theta^0 = 1, \quad \theta^{k+1} = (1 - \alpha^k \eta)\theta^k,$$

with

$$\begin{aligned} r_P^k &= \theta^k \ r_P^0, \\ r_D^k &= \theta^k \ r_D^0, \\ r_G^k &= \theta^k \ r_G^0. \end{aligned}$$

Proof. We have

$$\begin{aligned}
\mu^{k+1} &= [(x^{k+1})^T z^{k+1} + (\tau^{k+1})^T \kappa^{k+1}] \;/\; (n+1) \\
&= \{\; [(x^k)^T z^k + \tau^k \kappa^k] + \alpha^k \; [(x^k)^T d_z + (d_x)^T z^k + \tau^k d_\kappa + d_\tau \kappa^k] + (\alpha^k)^2 \; [(d_x)^T d_z + d_\tau d_\kappa] \;\} \;/\; (n+1) \\
&= [1 - \alpha^k(1-\gamma) + (\alpha^k)^2 \eta (1-\gamma-\eta)] \; \mu^k \\
&= (1-\alpha^k \eta)[1 - \alpha^k(1-\gamma-\eta)] \; \mu^k .
\end{aligned}$$

We also have

$$\begin{aligned}
r_P^{k+1} &= (\tau^k + \alpha^k d_\tau) b - A(x^k + \alpha^k d_x) \\
&= r_P^k + \alpha^k (d_\tau b - A d_x) \\
&= (1-\alpha^k \eta) r_P^k .
\end{aligned}$$

Similarly, we have this relation for r_D^{k+1} and r_G^{k+1} as well.

□

Lemma 4 . *The generic algorithm generates* $(y^k, x^k, \tau^k, z^k, \kappa^k, \mu^k, \theta^k)$ *satisfying*

$$(x^0)^T z^k + (z^0)^T x^k + \tau^0 \kappa^k + \kappa^0 \tau^k = [\; \frac{\mu^k}{\theta^k} + \theta^k \mu^0 \;] \; (n+1). \tag{11}$$

Proof. From (6), we have

$$[(y^k)^T r_P^0 - (x^k)^T r_D^0 + \tau^k r_G^0] \; \theta^k = \mu^k (n+1).$$

Thus

$$\begin{aligned}
\tfrac{\mu^k}{\theta^k}(n+1) &= (y^k)^T (b\tau^0 - Ax^0) - (x^k)^T (c\tau^0 - A^T y^0 - z^0) + \tau^k (c^T x^0 - b^T y^0 + \kappa^0) \\
&= [\tau^k c^T - (y^k)^T A] x^0 + [(y^k)^T b - (x^k)^T c] \tau^0 + [(x^k)^T A^T - \tau^k b^T] y^0 + (x^k)^T z^0 + \tau^k \kappa^0 \\
&= (x^0)^T [z^k + r_D^k] + \tau^0 [\kappa^k - r_P^k] + (y^0)^T (-r_P^k) + (z^0)^T x^k + \kappa^0 \tau^k \\
&= (x^0)^T z^k + (z^0)^T x^k + \tau^0 \kappa^k + \kappa^0 \tau^k - \theta^k [(y^0)^T r_P^0 - (x^0)^T r_D^0 + \tau^0 r_G^0] \\
&= (x^0)^T z^k + (z^0)^T x^k + \tau^0 \kappa^k + \kappa^0 \tau^k - \theta^k \mu^0 (n+1).
\end{aligned}$$

□

Let us elaborate this lemma a little bit. If μ^k and θ^k decrease at the same rate, i.e.,

$$\Omega(1) \le \frac{\mu^k}{\theta^k} \le O(1) \quad \text{and} \quad \theta^k \to 0, \tag{12}$$

then we can regard (11) as the normalizing constraint explicitly used in [15]. This makes $(x^k, z^k, \tau^k, \kappa^k)$ on a simplex. However, if $\mu^k/\theta^k \to 0$, then the iterates converge to the all-zero solution which is the origin of the cone (HLF). Furthermore, if $\mu^k/\theta^k \to \infty$, then we have diverging iterates. This behavior has been rigorously discussed by Mizuno et al. [11]. Nevertheless, while maintaining relations (9) and (12), we have some freedom in choosing η and γ in each iteration. In particular, with $\eta = 1-\gamma$ in all iterations, the generic algorithm becomes Ye-Todd-Mizuno's homogeneous and self-dual algorithm. This can be seen from Lemma 2 that

$$(d_x)^T d_z + d_\tau d_\kappa = 0,$$

which implies that $\mu^k/\theta^k = \mu^0$ for all k. Then, from Lemma 4

$$(x^0)^T z^k + (z^0)^T x^k + \tau^0 \kappa^k + \kappa^0 \tau^k = [1+\theta^k]\mu^0 \ (n+1),$$

and it is precisely the normalization constraint in the homogeneous and self-dual formulation of [15].

Note

$$\frac{\theta^{k+1}}{\theta^k} - \frac{\mu^{k+1}}{\mu^k} = \alpha^k(1-\alpha^k\eta)(1-\gamma-\eta).$$

If $\alpha^k\eta = 1$, then both θ^{k+1} and μ^{k+1} are zero, and optimality and feasibility for the model are achieved simultaneously. Suppose $\alpha^k\eta < 1$. Then, at iteration k we have

- if $\eta = 1-\gamma$, then both θ and μ (or both feasibility and optimality) decrease at the same rate, and $(d_x)^T d_z + d_\tau d_\kappa = 0$;
- if $\eta < 1-\gamma$, then μ decreases faster than θ (or optimality improves faster than feasibility) does, and $(d_x)^T d_z + d_\tau d_\kappa > 0$;
- if $\eta > 1-\gamma$, then θ decreases faster than μ (or feasibility improves faster than optimality) does, and $(d_x)^T d_z + d_\tau d_\kappa < 0$.

We can control the step size such that the iterates follow a family of central paths with parameter θ

$$\mathcal{C}(\theta) = \left\{ (y,\ x>0, \tau>0, z>0, \kappa>0) : \begin{pmatrix} Xz \\ \tau\kappa \end{pmatrix} = \mu e, \quad \begin{pmatrix} r_P \\ r_D \\ r_G \end{pmatrix} = \theta \begin{pmatrix} r_P^0 \\ r_D^0 \\ r_G^0 \end{pmatrix} \right\}.$$

Each limit point, under condition (12), is a strictly complementary point for (HLF), according to Mizuno et al. [11]. A simple choice ($x^0 = e, y^0 = 0, z^0 = e, \tau^0 = 1, \kappa^0 = 1$) ensures that the initial point is on path $\mathcal{C}(1)$. The algorithm generates iterates in a neighborhood of a two dimensional (μ,θ) surface given by

$$\mathcal{N}(\beta) = \left\{ (y,\ x, \tau, z, \kappa) : \left\| \begin{pmatrix} Xs \\ \tau\kappa \end{pmatrix} - \mu e \right\| \le \beta\mu, \quad \begin{pmatrix} r_P \\ r_D \\ r_G \end{pmatrix} = \theta \begin{pmatrix} r_P^0 \\ r_D^0 \\ r_G^0 \end{pmatrix}, \ \Omega(1) \le \frac{\mu}{\theta} \le O(1) \right\}$$

for some $\beta \in (0,1)$. The boundary property of this type of surface is discussed in Mizuno et al. [11].

Applying Mizuno-Todd-Ye's predictor-corrector technique [10] for (HLF), with

$$\gamma = 0, \quad \text{and} \quad \eta = 1 \quad (\text{or } 0 < \eta \le 1 + O(1/(n+1)) \text{ finite times}) \tag{13}$$

in the predictor step, and

$$\gamma = 1, \quad \text{and} \quad \eta = 0 \quad (\text{or } 0 < \eta \le O(1/\sqrt{n+1}) \text{ finite times}) \tag{14}$$

in the corrector step, we can verify that relations (9) and (12) hold and the algorithm will generate strictly complementary solution for (HLF) in $O(\sqrt{n}L)$ iterations (see the projection scheme of Mehrotra and Ye [9]).

In particular, if $\eta = 1$ and $\eta = 0$ in the predictor and corrector steps, respectively, for all iterations, then μ and θ decrease at the same rate, and the iterates follow the path

$$\mathcal{C} = \left\{ (x, y, z, \tau, \kappa,) : \begin{pmatrix} Xz \\ \tau\kappa \end{pmatrix} = \mu e, \quad \begin{pmatrix} r_P \\ r_D \\ r_G \end{pmatrix} = \theta \begin{pmatrix} r_P^0 \\ r_D^0 \\ r_G^0 \end{pmatrix}, \quad \theta = \frac{\mu}{\mu^0} \right\}.$$

In summary, we state the following theorem, whose proof is similar to that in [15].

Theorem 5 . *The homogeneous and self-dual algorithm generates a strictly complementary feasible solution, $(y^*, x^*, \tau^*, s^*, \kappa^*)$, for (HLF) in $O(\sqrt{n}L)$ iterations, by which we either obtain a strictly complementary optimal solution for (LP) and (LD) or detect the infeasibility for at least one of them (see* **iv** *and* **v** *of Theorem 1).*

4 Implementation

In this section, we describe an implementation of the homogeneous and self-dual algorithm specified in the last section with several minor modifications.

Using the same matrix factors for both predictor and corrector steps

In contrast to Mizuno-Todd-Ye [10], Mehrotra [8] solves both predictor and corrector steps in one iteration using the same matrix factors plus a second-order corrector term on the right-hand side. This technique saves time in the corrector steps and performs very well in practice. We adapt this technique in our implementation and solve the following system, coupled with system (3),

$$\begin{aligned} X^k d_z + Z^k d_x &= -X^k z^k, \\ \tau^k d_\kappa + \kappa^k d_\tau &= -\tau^k \kappa^k. \end{aligned}$$

Denote the resulting affine direction as $(d_x^a, d_y^a, d_z^a, d_\tau^a, d_\kappa^a)$. Using the standard nonnegativity ratio test along the direction, we obtain a step size α. Let

$$\mu_a = [\ (x^k + \alpha d_x^a)^T (z^k + \alpha d_z^a) + (\tau^k + \alpha d_\tau^a)(\kappa^k + \alpha d_\kappa^a)\ ]\ /\ (n+1).$$

From Lemma 3 we have

$$\frac{\mu_a}{\mu^k} = (1 - \alpha\eta)[1 - \alpha(1-\eta)\ ].$$

Then, we try using μ_a/μ^k to estimate the centering parameter γ for the corrector step.

Note that the residual of complementarity after solving the above system is

$$D_x^a d_z^a \quad \text{and} \quad d_\tau^a d_\kappa^a,$$

where $D_x^a = diag(d_x^a)$. Adding this residual as a corrector term and coupling with a properly estimated centering term, we solve the following system, combined with system (3) again,

$$\begin{aligned} X^k d_z + Z^k d_x &= -X^k z^k - D_x^a d_z^a + \gamma\mu^k e, \\ \tau^k d_\kappa + \kappa^k d_\tau &= -\tau^k \kappa^k - d_\tau^a d_\kappa^a + \gamma\mu^k. \end{aligned}$$

(Note that the system is not exactly the centering step since it contains $D_x^a d_z^a$ and $d_\tau^a d_\kappa^a$ on the right-hand side, see Mehrotra [7][8].) For system (3), we choose $\eta = 1 - \gamma$ in the corrector step and $\eta = 1$ in the predictor step.

Solving normal equation system

The major part of computational work in each iteration is concerned with solving a normal equation system

$$A\Theta A^T s = r.$$

This is accommodated by performing the Cholesky factorization of $A\Theta A^T$, yielding

$$A\Theta A^T = LDL^T,$$

where L is a lower triangular matrix with unitary diagonal element, and D is a positive diagonal matrix.

Before the factorization, a minimum degree heuristic is performed to determine a row permutation of A which attempts to minimize the number of nonzeros in L. The factorization procedure is written in FORTRAN 77, based on the framework of IPMOS, an interior-point method optimization system developed by Xu [14]. Memory allocation is carefully managed for exploiting cache memory.

During the factorization, if a pivot value is less than a machine related tolerance, then the current row i can be regarded as an almost linear dependent row in the current iteration. In our implementation, besides setting

$$D_{ii} = 1 \quad \text{and} \quad L_{ji} = 0, \text{ for } j > i,$$

we set

$$\bar{r}_i = 0$$

during forward substitution for solving $L\bar{r} = r$. Numerical experience shows that setting $\bar{r}_i = 0$ makes the algorithm more stable (E. Andersen, private communication). In fact, the proposed strategy is equivalent to ignoring column i and row i from $A\Theta A^T$ in the current iteration.

Handling homogeneous column and row

The introduction of homogeneous variable τ may result in both a dense column and a dense row in the Newton equation system. In our implementation, we treat that column and row separately. Overall, in each iteration we solve a system against three right-hand vectors with the same factors of $A\Theta^k A^T$, where $\Theta^k = X^k (Z^k)^{-1}$. Note that in the traditional implementation of primal-dual interior-point algorithms (with predictor-corrector strategy), the same system is solved against two right-hand vectors. No other dense columns of A are treated separately at this moment.

Treatment of bounded and free variables

NETLIB problems may have bounded and free variables. As a preliminary implementation, we simply consider upper bounds as explicit constraints and split free variables into a pair of positive and negative artificial variables. Usually, splitting a free variable into the difference of two new variables gives rise to numerical problems, because $A\Theta A^T$ becomes ill-conditioned when the pair go large. However, due to lemma 4 and setting $\eta = 1-\gamma$, the implicit normalizing constraint automatically prevents the pair of free variable from going large. Of course, with a more sophisticated technique to handle bounded and free variables, we expect a significant improvement in performance for our implementation.

Primal and dual step sizes

Numerical experience shows that properly taking different step sizes in primal and dual updates may enhance algorithm's performance [5]. We adapt this strategy in our implementation. Let

$$\begin{aligned}\alpha_x &= min\ \{\ -x_i/(d_x)_i,\ if\ (d_x)_i < 0\ \}, \\ \alpha_z &= min\ \{\ -z_i/(d_z)_i,\ if\ (d_z)_i < 0\ \}, \\ \alpha_\tau &= -\tau/d_\tau,\ if\ d_\tau < 0, \\ \alpha_\kappa &= -\kappa/d_\kappa,\ if\ d_\kappa < 0,\end{aligned}$$

and let

$$\begin{aligned}\alpha_P &= 0.99995 \times min\ \{\ \alpha_x, \alpha_\tau, \alpha_\kappa\ \}, \\ \alpha_D &= 0.99995 \times min\ \{\ \alpha_z, \alpha_\tau, \alpha_\kappa\ \}.\end{aligned}$$

Afterwards, we check to see if it is beneficial to take different step sizes, according to the following rule:

1. if $\alpha_P > \alpha_D$, then if $(z^k)^T d_x \le 0$, we take different step sizes;
2. if $\alpha_P < \alpha_D$, then if $(x^k)^T d_z \le 0$, we take different step sizes.

Otherwise, we use the same step size for both primal and dual updates.

Suppose different step sizes are taken. Then for the primal update

$$\begin{aligned}x^+ &= x^k + \alpha_P d_x, \\ \tau_P^+ &= \tau^k + \alpha_P d_\tau,\end{aligned}$$

and for the dual update

$$\begin{aligned}y^+ &= y^k + \alpha_D d_y, \\ z^+ &= z^k + \alpha_D d_z, \\ \tau_D^+ &= \tau^k + \alpha_D d_\tau.\end{aligned}$$

Since our feasibility model introduces a homogeneous variable τ for both b and c, moving in different step sizes for primal variable x and dual variable (y,z) causes different τ_P^+ and τ_D^+. In our implementation, we let the smaller one be the next τ^+, and κ^+ be its partner, i.e.

$$\begin{aligned}\tau^+ &= min\ \{\tau_P, \tau_D\}, \\ \kappa^+ &= \begin{cases} \kappa^k + \alpha_D d_\kappa & if \quad \tau^+ = \tau_P, \\ \kappa^k + \alpha_P d_\kappa & if \quad \tau^+ = \tau_D. \end{cases}\end{aligned} \tag{15}$$

Then update the iterate as

$$\begin{aligned}
\tau^{k+1} &= \tau^+, \\
\kappa^{k+1} &= \kappa^+, \\
x^{k+1} &= (\tau^+/\tau_P)\ x^+, \\
y^{k+1} &= (\tau^+/\tau_D)\ y^+, \\
z^{k+1} &= (\tau^+/\tau_D)\ z^+.
\end{aligned}$$

Choosing parameters γ and η

An important issue in the predictor-corrector algorithm is to choose a proper γ. We actually use the following rule: Let

$$\begin{aligned}
\mu^k &= [(x^k)^T z^k + \tau^k \kappa^k] \ / \ (n+1), \\
\mu_a &= [(x^k + \alpha_P d_x^a)^T (z^k + \alpha_D d_z^a) + (\tau^k + \alpha_\tau^+ d_\tau^a)(\kappa^k + \alpha_\kappa^+ d_\kappa^a)] \ / \ (n+1),
\end{aligned}$$

where α_τ^+ and α_κ^+ are the step sizes of d_τ^a and d_κ^a according to (15). Then, choose

$$\gamma = \begin{cases} (\mu_a/\mu^k)^2 & if \qquad \mu_a/\mu^k \le 10^{-2}, \\ min\{\ 10^{-1},\ max\{(\mu_a/\mu^k)^3,\ 10^{-4}\}\ \} & otherwise, \end{cases}$$

and let

$$\eta = 1 - \gamma.$$

The above choice keeps $\gamma \le 0.1$ and makes $\eta \ge 0.9$ which guarantees a large reduction of primal and dual infeasibility residuals when a long step is taken.

Stopping criteria

Our program terminates whichever of the following two criteria is satisfied:

1. Optimal (approximate) solution is obtained if

$$\begin{aligned}
\frac{|c^T x - b^T y|}{\tau + |b^T y|} &< 10^{-8} \\
\frac{||r_P||}{\tau + ||x||} &< 10^{-8} \\
\frac{||r_D||}{\tau + ||z||} &< 10^{-8}.
\end{aligned}$$

2. LP problem is infeasible (or near infeasible) if

$$\begin{aligned}
\mu/\mu^0 &< 10^{-14} \\
\frac{\tau}{\kappa} / \frac{\tau 0}{\kappa 0} &< 10^{-8}.
\end{aligned}$$

Stopping criterion 1 is independent of the size of τ. Letting

$$(x', y', z') = (x/\tau, y/\tau, z/\tau),$$

we have

$$((r_P)', (r_D)') = (r_P/\tau, r_D/\tau).$$

Hence

$$\frac{|c^T x - b^T y|}{\tau + |b^T y|} = \frac{|c^T x' - b^T y'|}{1 + |b^T y'|}, \quad \frac{||r_P||}{\tau + ||x||} = \frac{||(r_P)'||}{1 + ||x'||}, \quad \frac{||r_D||}{\tau + ||z||} = \frac{||(r_D)'||}{1 + ||z'||}.$$

Thus, for solving feasible problems, our stopping criterion is identical to that used in Lustig et al. [5] and Mehrotra [8].

For solving infeasible LP problems, we notice that τ^k/κ^k does decrease quadratically in our experiment, which correctly proves infeasibility in theory (Wu et al. [13]). We find that the use of criterion

$$\frac{\tau}{\kappa} / \frac{\tau^0}{\kappa^0} < 10^{-8}$$

or

$$\frac{\tau}{\kappa} / \frac{\tau^0}{\kappa^0} < 10^{-10}$$

ususally does not make a difference in running time.

Initial point

We choose the starting point simply as

$$x^0 = e, \quad y^0 = 0, \quad z^0 = e,$$
$$\tau^0 = 1, \quad \text{and} \quad \kappa^0 = 1.$$

Starting from this initial point and using the default values for program parameters, our algorithm solves or detects infeasibility of all encountered test problems. Our algorithm seems less sensitive to the selection of the initial point than OB1 or Mehrotra [8] does. Numerical experience shows that it would be difficult for them to solve some of the NETLIB feasible problems if starting from a simple initial point such as $x = e$, $y = 0$, $z = e$. Several papers considered various initial point selections, see Lustig et al. [4], Lustig et al. [5], Mehrotra [7], Mehrotra [8], Altman et al. [1], and etc. According to their descriptions, they generally need to solve a least-squares problem at the beginning to construct the initial point, where the amount of work is like one iteration.

However, we would like to point out that a good initial point choice could further improve the performance of our code by a small margin. For instance, starting same as Mehrotra [8] and without changing any other program parameters, our code solves problem 25FV47 in 25 iterations and problem 80BAU3B in 30 iterations. Thus, more efficient results are expected by looking into initial point selection for our code.

5 Preliminary computational results

In this section, computational results of running our implementation code, as well as OB1, on a APOLLO 720 workstation are reported. Both FORTRAN codes are compiled by the Unix Fortran compiler F77 with the highest optimization option $+O3$.

The reported cpu time (in seconds) of OB1 includes preprocessing, such as matrix cleanup, ordering and symbolic factorization, and arithmetic operations till the solver terminates. However, we exclude any efforts required by OB1 to translate files such as MPS input and output, either at the beginning or in the middle of the run. Default parameters are used (expect for WOOD1P and WOODW). The reported cpu time (in seconds) of our code includes all operations, except for MPS file input and output. All default parameters are used. All the NETLIB feasible problems are successfully solved except DFL001 and FIT2P. The sizes of these two problems are too large for our machine memory capacity.

Table I contains detailed computational results for solving NETLIB feasible problems. I_O and I_H represent the number of iterations used by OB1 and our code, called HLF, respectively; T_O and T_H are cpu times used by OB1 and HLF, respectively. I_{H-O} is the difference in the number of iterations between OB1 and HLF. The positive sign indicates that OB1 requires fewer iterations, while the negative sign means that HLF requires fewer. The last column is the cpu time comparison. Again, the positive rate represents how much OB1 is faster than HLF, and the negative rate indicates how much HLF is faster than OB1.

For several problems, such as FIT1D, FIT1P, FIT2D, our code takes much more time per iteration. This is because these problems have many variables with upper bounds. OB1 handles upper bounds implicitly and intelligently. But our technique, i.e, explicitly including upper bounds as constraints, results in a very big constraint matrix in contrast to the original A, and results in slow performance. For problems WOOD1P and WOODW, as suggested by Lustig (private communication), we reset "DEPENDENT NO" in the specification input file for OB1. Otherwise, due to the data structure of their coefficient matrix, it takes significant cpu time to perform Gauss elimination, which is employed by OB1.60 in order to detect dependent rows.

Table II includes computational results of both OB1 and HLF for solving the NETLIB infeasible problems. There are 29 problems in the set of NETLIB infeasible problems. Two of them, CERIA3D and CPLEX1, have dense columns of near to or more than one thousand nonzero entries, which causes space trouble for our code to handle $A\Theta A^T$. (The special dense column technique has not been employed in our current implementation.) For all other problems, infeasibility is successfully detected by HLF. Particularly, CPLEX2, an almost feasible problem, contributed by CPLEX Optimization Inc., is a very interesting problem for testing infeasibility detection capability. At iteration 15 of our run, both primal and dual infeasibility are less than 10^{-7}, and primal and dual objective values share the same first 6 digits. Finally, its infeasibility is detected in iteration 44. For solving this set of problems, OB1 seems to terminate with mixed answers ranging from 0: "crush(presolve) detects that model is infeasible", 1: "optimal", 3: "model appears to be dual unbounded. (or primal infeasible)", to 5: "numerical difficulties encountered. unable to reach solution with desired accuracy".

It is well known that preprocessing (or presolve) does an important job in many up-to-date LP codes such as a simplex code CPLEX and an interior-point code OB1. For instance, the presolver employed in OB1.60 can remove more than 1/3 rows and columns of problem AGG. However, almost no preprocessing is employed in our implementation at this moment. Thus, our results are very preliminary. By employing data preprocessing, as well as a better technique for bounded and free variables, a numerical procedure for dense columns, and a finite termination scheme for optimal bases, we expect a powerful and complete implementation for solving LP in the near future.

Acknowledgement We thank Professors Lustig, Marsten, and Shanno for providing us OB1.60 in this research. We also thank our colleagues E. Andersen, K. Anstreicher, K. Kortanek, F. Potra, and M. Todd for their helpful comments and suggestions.

References

[1] A. Altman and J. Gondzio An efficient implementation of a higher order primal-dual interior point method for large sparse linear programming, Technical Report ZTSW-1/A214/92, Systems Research Institute, Polish Academy of Sciences, Poland (1992).

[2] A. J. Goldman and A. W. Tucker, Polyhedral convex cones, in: *Linear Inequalities and Related Systems,* ed. H. W. Kuhn and A. W. Tucker (Princeton University Press, Princeton, NJ, 1956) p. 19.

[3] M. Kojima, N. Megiddo, and S. Mizuno, A primal-dual infeasible-interior-point algorithm for linear programming, Mathematical Programming 61(1993)263.

[4] I. J., Lustig, R. E. Marsten, and D. F. Shanno, Starting and restarting the primal-dual interior point method, Rutcor Research Report RRR # 61-90, Rutgers, New Brunswick, NJ (1990).

[5] I. J., Lustig, R. E. Marsten, and D. F. Shanno, On implementing Mehrotra's predictor-corrector interior point method for linear programming, SIAM Journal on Optimization 2(1992)435.

[6] I. J., Lustig, R. E. Marsten, and D. F. Shanno, Computational experience with a globally convergent primal-dual predictor-corrector algorithm for linear programming, Mathematical Programming 66(1994)123.

[7] S. Mehrotra Higher order methods and their performance, Technical Report, TR90-16R1 (revision July 1991), Department of IE and MS, Northwestern University, Evanston, IL (1991).

[8] S. Mehrotra On the implementation of a (primal-dual) interior point method, SIAM Journal on Optimization 2(1992)575.

[9] S. Mehrotra and Y. Ye Finding an interior point in the optimal face of linear programs, Mathematical Programming 62(1993)497.

[10] S. Mizuno, M. J. Todd, and Y. Ye, On adaptive-step primal-dual interior-point algorithms for linear programming, Mathematics of Operations Research 18(1993)964.

[11] S. Mizuno, M. J. Todd, and Y. Ye, A surface of analytic centers and infeasible-interior-point algorithms for linear programming, Mathematics of Operations Research 20(1995)135.

[12] A. W. Tucker, Dual systems of homogeneous linear relations, in: *Linear Inequalities and Related Systems,* ed. H. W. Kuhn and A. W. Tucker (Princeton University Press, Princeton, NJ, 1956) p. 3.

[13] F. Wu, S. Wu, and Y. Ye, On quadratic convergence of the homogeneous and self-dual linear programming algorithm, Working Paper, College of Business Administration, The University of Iowa, Iowa City, IA (1993).

[14] X. Xu, Interior Point Method for Linear Programming : Theory & Practice, Ph.D. Dissertation, Institute of Systems Science, Academia Sinica, Beijing, China (1991).

[15] Y. Ye, M. J. Todd, and S. Mizuno, An $O(\sqrt{n}L)$-iteration homogeneous and self-dual linear programming algorithm, Mathematics of Operations Research 19(1994)53.

Table I: Performance on Solving NETLIB Feasible Problems (A-F)

Problem Summary					OB1.60		HLF		Ratio		
Name	Rows	Cols	Nonzeros	BR	I_O	T_O	I_H	T_H	I_{H-O}	$max\{\frac{T_O}{T_H}, \frac{T_H}{T_O}\}^*$	
25FV47	822	1571	11127		22	12.52	30	14.96	8	1.19	+
80BAU3B	2263	9799	29063	B	29	33.46	35	31.17	6	1.07	−
ADLITTLE	57	97	465		14	.14	13	.11	-1	1.27	−
AFIRO	28	32	88		9	.07	7	.06	-2	1.17	−
AGG	489	163	2541		23	2.94	18	2.82	-5	1.04	−
AGG2	517	302	4515		18	4.59	18	4.27	0	1.07	−
AGG3	517	302	4531		17	4.36	17	4.00	0	1.09	−
BANDM	306	472	2659		19	1.30	19	.87	0	1.49	−
BEACONFD	174	262	3476		10	.52	10	.47	0	1.11	−
BLEND	75	83	521		12	.24	10	.14	-2	1.71	−
BNL1	644	1175	6129		25	5.14	35	5.19	10	1.01	+
BNL2	2325	3489	16124		32	58.45	34	63.25	2	1.08	+
BOEING1	351	384	3865	BR	24	2.29	26	2.61	2	1.14	+
BOEING2	167	143	1339	BR	14	.47	18	.60	4	1.28	+
BORE3D	234	315	1525	B	16	.54	16	.53	0	1.02	−
BRANDY	221	249	2150		17	.87	18	.71	1	1.23	−
CAPRI	272	353	1786	B	18	1.43	17	1.17	-1	1.22	−
CYCLE	1904	2857	21322	B	28	32.86	29	39.14	1	1.19	+
CZPROB	930	3523	14173	B	31	6.83	37	5.27	6	1.30	−
D2Q06C	2172	5167	35674		32	157.21	40	153.01	8	1.03	−
D6CUBE	416	6184	43888	B	21	36.41	14	24.57	-7	1.48	−
DEGEN2	445	534	4449		14	3.84	11	2.38	-3	1.61	−
DEGEN3	1504	1818	26230		19	76.03	16	58.02	-3	1.31	−
E226	224	282	2767		21	1.15	18	.69	-3	1.67	−
ETAMACRO	401	688	2489	B	26	4.06	27	3.66	1	1.11	−
FFFFF800	525	854	6235		28	6.97	29	6.54	1	1.07	−
FINNIS	498	614	2714	B	20	1.69	26	1.54	6	1.10	−
FIT1D	25	1026	14430	B	15	1.87	16	4.67	1	2.50	+
FIT1P	628	1677	10894	B	16	157.01	18	369.05	2	2.35	+
FIT2D	26	10500	138018	B	18	28.22	21	172.66	3	6.12	+

* $-$ indicates $T_H/T_O < 1$ and $+$ indicates $T_O/T_H < 1$. Also $I_{H-O} = I_H - I_O$.

Table I: Performance on Solving NETLIB Feasible Problems (G-S)

Problem Summary					OB1.60		HLF		Ratio	
Name	Rows	Cols	Nonzeros	BR	I_O	T_O	I_H	T_H	I_{H-O}	$max\{\frac{T_O}{T_H}, \frac{T_H}{T_O}\}^*$
FORPLAN	162	421	4916	BR	21	1.53	27	1.52	6	1.01 −
GANGES	1310	1681	7021	B	16	7.49	17	5.99	1	1.25 −
GFRD-PNC	617	1092	3467	B	19	1.52	17	.82	-2	1.85 −
GREENBEA	2393	5405	31499	B	41	39.16	46	54.32	5	1.39 +
GREENBEB	2393	5405	31499	B	33	31.46	44	49.90	11	1.59 +
GROW15	301	645	5665	B	12	1.73	14	1.94	2	1.12 +
GROW22	441	946	8318	B	12	2.62	15	3.13	3	1.19 +
GROW7	141	301	2633	B	10	.71	12	.75	2	1.06 +
ISRAEL	175	142	2358		25	3.86	20	3.66	-5	1.05 −
KB2	44	41	291	B	15	.17	16	.10	1	1.70 −
LOTFI	154	308	1086		16	.61	15	.30	-1	2.03 −
MAROS	847	1443	10006	B	30	7.66	27	8.28	-3	1.08 +
NESM	663	2923	13988	BR	31	13.24	34	13.51	3	1.02 +
PEROLD	626	1376	6026	B	36	13.52	35	12.26	-1	1.10 −
PILOT	1442	3652	43220	B	29	209.06	44	409.26	15	1.96 +
PILOT.JA	941	1988	14706	B	43	42.72	38	34.71	-5	1.23 −
PILOT.WE	723	2789	9218	B	37	11.70	40	10.11	3	1.16 −
PILOT4	411	1000	5145	B	35	8.91	37	8.41	2	1.06 −
PILOT87	2031	4883	73804	B	41	1007.14	56	1778.40	15	1.77 +
PILOTNOV	976	2172	13129	B	20	19.59	23	24.61	3	1.26 +
RECIPE	92	180	752	B	9	.18	11	.13	2	1.38 −
SC105	106	103	281		10	.18	10	.08	0	2.25 −
SC205	206	203	552		11	.29	11	.15	0	1.93 −
SC50A	51	48	131		9	.10	9	.04	0	2.50 −
SC50B	51	48	119		8	.08	8	.03	0	2.67 −
SCAGR25	472	500	2029		17	.99	16	.45	-1	2.20 −
SCAGR7	130	140	553		15	.22	13	.14	-2	1.57 −
SCFXM1	331	457	2612		17	1.32	20	.92	3	1.43 −
SCFXM2	661	914	5229		19	3.07	22	2.24	3	1.37 −
SCFXM3	991	1371	7846		20	4.96	23	3.70	3	1.34 −
SCORPION	389	358	1708		12	.49	12	.34	0	1.44 −

Table I: Performance on Solving NETLIB Feasible Problems (S-W)

Problem Summary					OB1.60		HLF		Ratio			
Name	Rows	Cols	Nonzeros	BR	I_O	T_O	I_H	T_H	I_{H-O}	$max\{\frac{T_O}{T_H}, \frac{T_H}{T_O}\}^*$		
SCRS8	491	1169	4029		19	2.21	26	1.69	7	1.31	−	
SCSD1	78	760	3148		9	.42	9	.23	0	1.83	−	
SCSD6	148	1350	5666		10	.93	11	.53	1	1.75	−	
SCSD8	398	2750	11334		9	2.00	9	1.01	0	1.98	−	
SCTAP1	301	480	2052		14	.73	16	.43	2	1.70	−	
SCTAP2	1091	1880	8124		12	2.90	12	2.38	0	1.22	−	
SCTAP3	1481	2480	10734		15	4.84	13	3.24	-2	1.49	−	
SEBA	516	1028	4874	BR	18	23.93	26	51.78	8	2.16		+
SHARE1B	118	225	1182		20	.51	25	.36	5	1.42	−	
SHARE2B	97	79	730		11	.20	10	.14	-1	1.43	−	
SHELL	537	1775	4900	B	20	2.38	20	1.29	0	1.84	−	
SHIP04L	403	2118	8450		12	1.78	13	1.14	1	1.56	−	
SHIP04S	403	1458	5810		13	1.31	14	.82	1	1.60	−	
SHIP08L	779	4283	17085		14	3.72	18	3.35	4	1.11	−	
SHIP08S	779	2387	9501		14	1.80	19	1.95	5	1.08		+
SHIP12L	1152	5427	21597		16	5.38	24	5.62	8	1.04		+
SHIP12S	1152	2763	10941		14	2.29	22	2.62	8	1.14		+
SIERRA	1228	2036	9252	B	15	4.52	17	5.00	2	1.11		+
STAIR	357	467	3857	B	14	3.13	16	3.24	2	1.04		+
STANDATA	360	1075	3038	B	12	1.18	14	.63	2	1.87	−	
STANDGUB	362	1184	3147	B	12	1.21	14	.73	2	1.66	−	
STANDMPS	468	1075	3686	B	18	2.08	16	1.15	-2	1.81	−	
STOCFOR1	118	111	474		16	.31	12	.15	-4	2.07	−	
STOCFOR2	2158	2031	9492		25	11.49	30	7.99	5	1.44	−	
STOCFOR3	16676	15695	74004		34	164.21	48	113.14	14	1.45	−	
TRUSS	1001	8806	36642		18	19.36	18	19.08	0	1.01	−	
TUFF	334	587	4523	B	19	2.55	19	1.97	0	1.29	−	
VTP.BASE	199	203	914	B	13	.21	16	.41	3	1.95		+
WOOD1P	245	2594	70216		17	*12.45	14	18.36	-3	1.47		+
WOODW	1099	8405	37478		23	*18.54	26	23.45	3	1.26		+
total	91 problems				1753		1922		169		63	28

* We reset “DEPENDENT NO” in the OB1 specification input file.

Table II: Performance on Solving NETLIB Infeasible Problems (B-W)

Problem Summary					HLF		OB1.60		
Name	Rows	Cols	Nonzeros	BR	it	cpu	it	cpu	status*
bgdbg1	349	407	1485	B	6	.21			0
bgetam	401	688	2489	B	7	1.07	7	1.22	3
bgindy	2672	10116	75019		10	83.20	6	70.64	3
bgprtr	21	34	90		6	.02	9	.07	3
box1	232	261	912	B	5	.13	3	.15	1
chemcom	289	720	2190	B	5	.30	6	.56	3
cplex2	225	221	1059	B	44	1.44	47	1.81	1
ex72a	198	215	682	B	6	.12	10	.26	1
ex73a	194	211	668	B	6	.08	7	.19	5
forest6	67	95	270	B	6	.02	11	.10	3
galenet	9	8	16	B	4	.00			0
gosh	3793	10733	97257	B	23	224.83	17	107.13	3
gran	2569	2520	20151	B	17	167.30			0
greenbea**	2505	5405	35159	B	18	23.46			0
itest2	10	4	17		4	.04			0
itest6	12	8	23		5	.03			0
klein1	55	54	696		15	.25	54	.97	3
klein2	478	54	4585		16	54.81	38	67.21	3
klein3	995	88	12107		17	630.52	93	1548.14	3
mondou2	313	604	1623	B	8	.34	4	.27	3
pang	362	460	2666	B	24	1.80	33	3.04	5
pilot4i	411	1000	5145	B	15	3.76			0
qual	324	464	1714	B	32	1.70	59	4.22	5
reactor	319	637	2995	B	15	1.28			0
refinery	324	464	1694	B	13	.71	14	1.07	3
vol1	324	464	1714	B	30	1.63	42	3.14	3
woodinfe	36	89	209	B	10	.07			0

* OB1 termination status:

`0 crush(presolve) detects that model is infeasible.`

`1 optimal.`

`3 model appears to be dual unbounded. (or primal infeasible.)`

`5 numerical difficulties encountered. unable to reach solution with desired accuracy.`

** greenbea is the original version of the NETLIB infeasible GREENBEA problem. The feasible NETLIB version was created by Bob Fourer (Northwestern University) by repairing the version given here. The GREENBEA problem is a network model and originates in the petrochemical industry.