On Finding a Vertex Solution Using Interior Point Methods*

Sanjay Mehrotra†
Department of Industrial Engineering and Management Sciences
Northwestern University
Evanston, Illinois 60208

Submitted by Richard Tapia

ABSTRACT

An approach is proposed to generate a vertex solution while using a primal-dual interior point method to solve linear programs. A controlled random perturbation is made to the cost vector. A method to identify the active constraints at the vertex to which the solutions are converging is given. This basic method is further refined to save computational effort. The proposed approach is tested by using a variation of the primal-dual interior point method. Our method is developed by taking a predictor-corrector approach. In practice this method takes considerably fewer iterations to solve linear programs than methods described by Choi, Monma, and Shanno; Lustig, Marsten, and Shanno; and Domich, Boggs, Donaldson, and Witzgall. Computational results on problems from the NETLIB test set are reported to test our approach for finding vertex solutions. These results show that one perturbation is enough to force the solutions to converge to a vertex. The results indicate that the proposed approach is insensitive to the number of degenerate variables. The results also indicate that the effort required to generate a vertex solution is comparable to that required to solve the problem using an interior point method.

1. INTRODUCTION

Let us consider the linear program

$$P(0): \qquad \text{minimize} \quad z = c^T x$$
$$\text{s.t.} \quad Ax = b,$$
$$x \geq 0,$$

*This research was supported in part by the National Science Foundation grant CCR-8810107 and Office of Naval Research grant N00014-87-K-0214.

†E-mail: mehrotra@iems.nwu.edu.

655 Avenue of the Americas, New York, NY 10010 0024-3795/91/$3.50

where $c, x \in \Re^n$, $b \in \Re^m$, and $A \in \Re^{m \times n}$. Let z^* denote the optimal objective value of $P(0)$. We assume that $P(0)$ has a feasible solution and z^* is finite. Assume that A has full row rank and all columns of A have at least one nonzero element. The assumptions on A can be ensured in the beginning.

We are concerned with the problem of finding a vertex solution $\tilde{v}^*$ of $P(0)$ satisfying

$$c^T \tilde{v}^* - z^* \leqslant \epsilon \tag{1.1}$$

for a prespecified tolerance ϵ. Our emphasis here is that this problem be solved, as much as possible, within the framework of implementations of interior point methods. In the problem definition, if the choice of ϵ is large, then a vertex solution satisfying (1.1) is not necessarily optimal. On the other hand, if the problem data are rational, then for an appropriate (small) choice of ϵ, a vertex solution satisfying (1.1) is optimal [16, Lemma 8.6, p. 172].

The problem is of interest because it is known that the solutions obtained at the termination of interior point methods are in the interior of the optimal facet, unless some post processing is done. The known practical approaches to finding a vertex from an interior solution perform computations similar to those in the simplex method [5, 8]. It is now established that on a large class of real world problems interior point methods are competitive and often superior to the simplex method by a large factor (see e.g. [1, 2, 8, 11, 14]). For such problems there are potential disadvantages in using an approach based on the simplex method to find vertex solutions. The work involved in the simplex method is unpredictable and can be a large fraction of the work involved in solving the original problem by the simplex method. This can be true particularly if the problem is highly degenerate. This was indicated by the results reported in Marsten et al. [8].

The approach proposed in this paper makes a controlled random perturbation to the cost vector to improve the likelihood that the perturbed linear program has unique solution (at a vertex). The size of the perturbation is controlled so that the objective value at the optimal solution of the perturbed problem is within the specified tolerance of z^*. It is obvious that the proposed approach is probabilistic in nature. No probabilistic analysis of it is given. However, computational results on a large number of test problems are given to show that the proposed approach is highly reliable.

We emphasize that because of its probabilistic nature (and for numerical and algorithmic reasons) the proposed approach should not be viewed as a replacement for simplex type approaches. Instead its use should be considered for significantly reducing (possibly to zero) the work a simplex type approach would perform.

We used a variation of the primal-dual method for our computational testing. The motivation we provide here for this method introduces certain concepts of prediction and correction, and therefore we call it a predictor-corrector method. The proposed method requires considerably fewer iterations to solve linear programs.

This paper is organized as follows.

Section 2 develops the proposed approach of using random perturbations. We also discuss our reasons for proposing this approach.

Section 3.1 develops an approach to identify the constraints active at the vertex to which the solutions are assumed to be converging. Sections 3.2 gives modifications to the basic procedure to save computational effort in its implementation. Section 4 develops our primal-dual predictor-corrector method. Computational results on the vertex finding problem using a subset of NETLIB problems are discussed in Section 5.

NOTATION. The following notation is used in this paper. A vector inequality $x \geqslant 0$ is always understood componentwise. $\text{nz}(x)$ denotes the number of nonzero elements in x. Similarly, $\text{nz}(D)$ denotes the number of nonzero elements in the matrix D. $\text{rank}(A)$ represents the algebraic rank of the matrix A. e is used to represent a vector of all ones. $\| \ \|$ is used to represent the 2-norm of a vector. We use X to represent a diagonal matrix whose nonzero elements are the components of x.

2. ENCOURAGING VERTEX SOLUTIONS

Let $r \in \Re^n$ and $r \geqslant 0$. Consider the linear program

$$P(r): \qquad \begin{aligned} \text{minimize} \quad & z = (c + r)^T x \\ \text{s.t.} \quad & Ax = b, \\ & x \geqslant 0, \end{aligned}$$

which is the same as $P(0)$ for $r = 0$. The dual of $P(r)$ is given by

$$D(r): \qquad \begin{aligned} \text{maximize} \quad & b^T \pi \\ \text{s.t.} \quad & A^T \pi + s = c + r, \\ & s \geqslant 0. \end{aligned}$$

The following proposition is central to the development of our approach.

PROPOSITION 2.1. *Let $\hat{x}$ be a feasible solution for the linear program $P(0)$ (and $P(r)$). If $x^*(r)$ is an optimal solution of $P(r)$, then*

$$0 \leqslant c^T x^*(r) - z^* \leqslant (c + r)^T \hat{x} - z^*. \tag{2.1}$$

Proof. The lower bound follows by the definition of z^*. Now, $c^T x^*(r) - z^* \leqslant (c + r)^T x^*(r) - z^*$ because r and $x^*(r)$ are nonnegative vectors. The upper bound in (2.1) follows from the definition of $x^*(r)$. ∎

We now discuss the consequences of Proposition 2.1. If in addition to $\hat{x}$ a feasible solution $\hat{\pi}, \hat{s}$ of $D(0)$ is also known, then from (2.1) it is easy to see that

$$0 \leqslant c^T x^*(r) - z^* \leqslant \hat{s}^T \hat{x} + r^T \hat{x}. \tag{2.2}$$

The inequality (2.2) is important because it gives an *a priori* bound on the difference between z^* and the objective value of $x^*(r)$. It immediately suggests an approach to the vertex finding problem: $P(0)$ and $D(0)$ are first solved to a desired accuracy (say 0.1ϵ); then the cost vector of $P(0)$ is perturbed by r to obtain $P(r)$ and $D(r)$. Here r is chosen so that $\hat{s}^T \hat{x} + r^T \hat{x} < \epsilon$ and it is highly probable for $P(r)$ to have a vertex solution.

Although (2.2) suggest an approach, the requirement that feasible solutions $\hat{x}, \hat{\pi}, \hat{s}$ be known is restrictive. Experience indicates that implementations that approach primal and dual feasibility together with optimality are more efficient than those that first obtain feasibility and then approach optimality. Furthermore, in practice maintaining primal feasibility is difficult because of numerical errors. However, note that after a dual feasible solution is found, dual feasibility is easy to maintain in a primal-dual method. This is because of the form of dual constraints. Our unwillingness to assume the availability of primal and dual feasible solutions complicates the matter. We spend the rest of this section developing ways to deal with this.

If primal and dual feasible solutions are not available, then an inequality like (2.2) is not possible. But (2.1) still ensures

$$0 \leqslant c^T x^*(r) - z^* \leqslant r^T x^*, \tag{2.3}$$

where x^* is any optimal solution of $P(0)$. x^* is not known. Note, however, that in order to bound $r^T x^*$ from above we only need to know a bound on x_i^* for $i = 1, \ldots, n$, or on $\sum_{i=1}^n x_i^*$. In theory, such bounds are available provided we assume that the input data are rational and z^* is finite. In our context use of the theoretically available bound does not result in a practical

approach. This is because the theoretical bounds on x_i^* are very weak. Similar inefficiencies may arise if user provided weak bounds on x_i^* are used. In case a good bound on x_i^* is available, it may be used to generate $P(r)$ from the very first iteration.

Otherwise, if an algorithm is generating $\{x^k\} \to x^*$, then after some large k, x^k provides a good estimate of x^* and therefore of $r^T x^*$. This is the essence of the next proposition. Once again, $r^T x^* < \epsilon$ is ensured.

PROPOSITION 2.2. *Assume that an interior point algorithm generates* $\{x^k\} \to x^*$. *Let* $r_i = \epsilon \rho_i / 4n x_i^k$, $1 \leq \rho_i \leq 2$, *and* $\epsilon > 0$. *Then, for some large* k

$$r^T x^* \leq \epsilon. \tag{2.4}$$

Proof. Note that

$$r^T x^* = \frac{\epsilon}{4n} \sum_{i=1}^{n} \frac{\rho_i x_i^*}{x_i^k} \leq \frac{\epsilon}{2n} \sum_{x_i^* \neq 0} 1 + \frac{x_i^* - x_i^k}{x_i^k}. \tag{2.5}$$

Since $\{x^k\} \to x^*$, given any $\delta > 0$ there exists a $k(\delta)$ such that for all $k > k(\delta)$, $|x_i^* - x_i^k| < \delta$. Let us take $\delta = 0.5 \min\{x_i^* \mid x_i^* > 0\}$. Then, if $x_i^* \neq 0$, for all $k > k(\delta)$ we have $x_i^k > \delta$. Hence, (2.4) follows from (2.5). ■

Proposition 2.2 reduces the difficulty of knowing a bound on x^* to that of knowing an iteration $k(\delta)$ at which the estimates of the solution can be used reliably. At this iteration a perturbation vector r is generated as follows:

$$r_i = \frac{\epsilon}{40n} \frac{\text{RAND}(1,2)}{x_i^k}. \tag{2.6}$$

Here RAND(1,2) is a function generating a random variable which is uniformly distributed in (1,2). For a randomly generated r as in (2.6) we expect that $P(r)$ will almost always have a unique solution. This is supported by the computational evidence given in Section 5. In case the first perturbation fails to generate a vertex solution, we can make additional perturbations provided that the sum of all perturbations satisfies (2.4).

Using the discussion so far, a procedure which encourages the solutions to converge to a vertex within a specified tolerance can be developed provided that we have (i) a good heuristic for deciding when to generate $P(r)$, and (ii) a way to explicitly ensure that $c^T x^*(r) - z^* < \epsilon$. Here (ii) is needed because we must use a heuristic to decide the iteration at which r is

generated. There are several general approaches one may take to deal with this. We discuss three of these approaches and their relative merits next:

(A1) find an optimal solution of $P(0)$ and/or $D(0)$ in order to know z^* and generate a perturbed problem using this optimal solution;

(A2) generate $P(r)$ at an appropriately guessed iteration and use a simplex type approach to obtain an optimal vertex solution from the solution of $P(r)$;

(A3) generate a lower bound on z^* and use it to verify the stopping criterion for the vertex solution.

(A1): Solving $P(0)$ and $D(0)$ to optimality reduces the problem to the case where (2.2) can be used directly to generate r. The problem with using this approach is that it requires us to find an exact optimal solution of $P(0)$ and/or $D(0)$. This would involve processing an interior solution to generate an optimal solution. It is not clear how this could be done efficiently within the framework of interior point methods.

(A2): Since the approach being developed is based on a random perturbation, we do need to have a simplex type approach to ensure that a vertex solution is always found. However, the potential disadvantage in using it to verify $c^T x^*(r) - z^* \leqslant \epsilon$ is that in this case we must solve the problem till a primal and dual feasible basis for $P(0)$ and $D(0)$ is found in the simplex method. This could take many iterations for problems with highly degenerate primal optimal set.

(A3): This approach would be practical if lower bounds on z^* could be generated. A lower bound on z^* is provided by dual feasible solutions. Primal-dual algorithms keep estimates of both primal and dual solutions at each iteration. As mentioned before, after a dual feasible solution is obtained, maintaining dual feasibility is easy in these algorithms. But we are not assured a dual feasible solution in the implementations which do not require dual feasibility. This will be true particularly for problems with empty dual interior. However, in our experience we find that near-feasible dual solutions become available. In this case, if we know an M ensuring $x_i^* \leqslant M$, $i = 1, 2, \ldots, n$, for some x^*, then these solutions can be used to generate a lower bound on z^*. This is seen from Proposition 2.3 below. In this context the performance of the procedure with weak choices of M is quite satisfactory.

PROPOSITION 2.3. *Let $x^k > 0$ and $s^k > 0$, π^k estimate solutions of $P(0)$ and $D(0)$. Let $\xi_x^k \equiv Ax^k - b$ and $\xi_s^k \equiv A^T\pi^k + s^k - c \geqslant 0$. If x^* is an optimal solution for $P(0)$ satisfying $x_i^* \leqslant M$ for $i = 1, \ldots, n$, then*

$$c^T(x^k - x^*) \leqslant \pi^{k\,T}\xi_x^k + s^k - \xi_s^k + M\sum_{i=1}^{n} \xi_s^k \equiv \beta^k \tag{2.7}$$

and

$$z^* \geqslant b^T\pi^k - M\sum_{i=1}^{n} \xi_s^k \equiv \underline{z}^k. \tag{2.8}$$

Proof.

$$\begin{aligned} c^T(x^k - x^*) &= \left(A^T\pi^k + s^k - \xi_s^k\right)^T(x^k - x^*) \\ &= \pi^{kT}(Ax^k - b) + \left(s^k - \xi_s^k\right)^T x^k - s^{kT}x^* + \xi_s^{kT}x^* \\ &\leqslant \pi^{kT}\xi_x^k + \left(s^k - \xi_s^k\right)^T x^k + M\sum_{i=1}^{n} \xi_s^k. \end{aligned}$$

The last inequality follows because s^k, ξ_s^k, and x^* are nonnegative vectors and $x_i^* \leqslant M$. To see (2.8) introduce an additional constraint $\sum_{i=1}^{n}(\xi_s^k)_i x_i \leqslant M\sum_{i=1}^{n}(\xi_s^k)_i$ to obtain a problem $\hat{P}(0)$ from $P(0)$. The optimal objective value for $P(0)$ and $\hat{P}(0)$ is the same. (2.8) follows because $(\pi^{kT}, -1)^T$ is a feasible solution for the dual of $\hat{P}(0)$. ■

In the case where a satisfactory choice of M is not available, a dual feasible point can be generated by finding a near-optimal dual vertex. This can be done by generating a perturbed problem:

$$\begin{aligned} P'(r): \qquad & \text{minimize} \quad c^T x \\ & \text{s.t.} \quad Ax = b, \\ & \qquad x \geqslant -r. \end{aligned}$$

r used to generate $P'(r)$ is obtained from (2.6) with x_i replaced by s_i. The primal solution of $P(r)$ and the dual solution of $P'(r)$ can now be used to ensure the termination criterion.

We propose the use of (A3) in the current context. β^k in (2.7) is used to decide the iteration at which $P(r)$ is generated.

We summarize the discussion in this section in procedure ENCVERT (Table 1). In procedure ENCVERT, interior point methods are used for finding solutions of $P(0)$, $D(0)$, $P(r)$, and $D(r)$. The solutions of $P(0)$ and $D(0)$ from step 1 are used to get a starting point for $P(r)$ and $D(r)$. $x^k, \pi^k, s^k + r$ is taken as a starting point in step 3. The procedure MGTVERT (or GTVERT) is

TABLE 1

A PROCEDURE TO ENCOURAGE VERTEX SOLUTIONS

Procedure ENCVERT.

Step 1. Starting from x^0, **obtain** x^k satisfying $|\beta^k| \leq 0.1\epsilon$ for $P(0)$.
Step 2. **Obtain** r from (2.6).
Step 3. **Find** a vertex solution $v^*(r)$ of $P(r)$ [and $D'(r)$ if needed].
Step 4. If no vertex solution was found in step 3, either perform additional perturbations, or proceed with simplex type method; otherwise **goto** step 5.
Step 5. **If** $(c^T v^*(r) - \underline{z}^k \leq \epsilon)$ **EXIT**
Step 6. **Otherwise** $\epsilon \leftarrow \epsilon/10$; $x^0 \leftarrow x^k$, $\pi^0 \leftarrow \pi^k$, $s^0 \leftarrow s^k$; $k \leftarrow 0$; **goto** *step* 1.

used to jump to a vertex solution from an interior point. If the vertex solution of $P(r)$ does not satisfy the termination criterion, then we require more accurate solutions at step 1 before perturbing the problem.

3. EXTRACTING A VERTEX SOLUTION

This section assumes that the solution of the linear program considered is at a vertex v. By definition v is a feasible vertex iff the columns of A corresponding to positive components of v are linearly independent [15, p. 81]. A procedure is developed to obtain v from an interior point.

3.1. *Development of the Basic Procedure*

Let $\{x^k\}$ be a sequence of points (generated by an interior point method) converging to a vertex solution v, which is the optimal solution of $P(r)$. The following proposition is central to the development of our procedure. It essentially says that if $\{x^k\} \to v$, then the values of variables must partition into two sets: A set of variables having "large values" and a set of variables having "small values." The set with the small values gives the nonnegativity constraints active at v. Once this set is correctly determined, the values of variables positive at the vertex can also be determined.

PROPOSITION 3.1. *Let* $\{x^k\}$ *and* v *be as described above. Then*:

(i) *There exist a* δ *and* $k(\delta)$ *ensuring that* $\|x^k - v\| \leq \delta$ *for all* $k \geq k(\delta)$. *Furthermore, if* $v_i = 0$, *then* $x_i^k \leq \delta$; *otherwise* $x_i^k > \delta$.

(ii) *Let* $\|x^k - v\| \leq \delta$, *and* $\tilde{D}^k$ *be a diagonal matrix with*

$$\tilde{D}^k_{ii} = \begin{cases} 0 & \text{if } x^k_i \leq \delta \\ x^k_i & \text{otherwise,} \end{cases}$$

and $\tilde{B}^k \equiv A\tilde{D}^k$. *Then* $\operatorname{rank}(\tilde{B}^k) = \operatorname{nz}(v)$.

(iii) $v = \tilde{D}^k u$, *where* u *is a solution of* $\tilde{B}^k u = b$.

Proof. Since $\{x^k\} \to v$, for all δ there exist $k(\delta)$ such that $\|x^k - v\| \leq \delta$ for all $k \geq k(\delta)$. Now take $\delta = 0.1 \min\{v_i \mid v_i > 0\}$. If $\|x^k - v\| \leq \delta$, then for all i for which $v_i = 0$, we have $x_i \leq \delta$, and for all i for which $v_i > 0$, we have $x^k_i \geq v_i - \delta \geq 9\delta$.

The fact that columns of A corresponding to nonzero variables in v are linearly independent implies (ii). Since columns of $\tilde{B}^k$ corresponding to nonzero variables in v are linearly independent, b is obtained from a unique linear combination of these columns. $v_i / \tilde{D}^k_{ii}$ for $v_i > 0$ gives this combination. Hence, (iii) follows. ■

To know the correct partition of positive and zero variables at v, (i) in Proposition 3.1 requires us to know the smallest positive value a variable takes at v. A value for this can be obtained by using the input length of the problem. However, the value based on input length is not practical. We propose to search for the actual value. A search procedure will require a termination criterion. (ii) and (iii) in Proposition 3.1 can be used for that purpose. This is what we do in Procedure GTVERT given in Table 2.

If $\{x^k\}$ is sufficiently close to v, then from Proposition 3.1 and the above discussion we know that procedure GTVERT will generate v. However, since in practice we do not know the iteration at which x^k has come sufficiently close to v, it is possible that this procedure will return some other feasible vertex of the primal polytope. We can protect ourselves against this possibility by generating dual slacks, satisfying complementary slackness, corresponding to this vertex. If the vertex returned from GTVERT is optimal, then the slacks must be nonnegative.

It is a simple linear algebra exercise to show that all the computations required to implement GTVERT can be performed while computing a Cholesky factor of $A(\tilde{D}^k)^2 A^T$. Our implementation used the procedure in Mehrotra [11, Section 2] with minor modifications.

3.2. *Practical Modifications to* GTVERT

The main computational effort in implementing GTVERT comes from step 2. In this section we describe some useful modifications to GTVERT to make it more efficient.

TABLE 2

A PROCEDURE TO JUMP TO A VERTEX SOLUTION

Procedure GTVERT

Input: δ, x^k, A.

Step 1. **Let** $\tilde{B}^k = A\tilde{D}^k$, where

$$\tilde{D}^k_{ii} = \begin{cases} 0 & \text{if } x^k_i \leq \delta^k, \\ x^k_i & \text{otherwise.} \end{cases}$$

Step 2. **Determine** rank($\tilde{B}^k$).

If (rank($\tilde{B}^k$) = nz($\tilde{D}^k$)) **then**

Check if $\tilde{B}^k u = b$ has a solution.

Case I: **If** $\tilde{B}^k u = b$ has no solution, **then**

$\delta^k \leftarrow 0.1\delta^k$ **goto** step 1

endif

Case II: **If** $\tilde{B}^k u = b$ has a solution with negative components, **then**

Exit; try GTVERT at a later iteration

else

Exit; $\tilde{D}^k u$ is a vertex solution

endif

else

Exit; try GTVERT at a later iteration

endif

MODIFICATION 1. The procedure GTVERT described as in Section 3.1 computes a Cholesky factor at least once at each iteration. This is in addition to the computations in the main interior point algorithm. The Cholesky factor computed in GTVERT is first used to determine if rank $(\tilde{B}^k) = \text{nz}(\tilde{D}^k)$. The first modification comes from the observation that if we know in advance that

$$\text{rank}(\tilde{B}^k) < \text{nz}(\tilde{D}^k),$$

then there is no need for further computations in GTVERT. We keep information so that an upper bound on rank($\tilde{B}^k$) is available. This is accomplished by using Proposition 3.2. Clearly, if $\text{nz}(\tilde{D}^k) > m$, then there is no need to perform computations, since rank $(\tilde{B}^k) \leq m$.

PROPOSITION 3.2. *Let* $\hat{k}$, $\hat{k} < k$, *be the last iteration at which step* 2 *in* GTVERT *was performed. Let* $\text{nz}(\tilde{D}^{\hat{k}}\tilde{D}^k)$ *be the number of positive elements in*

$\tilde{D}^k$ which were zero in $\tilde{D}^{\hat{k}}$. Then

$$\operatorname{rank}(\tilde{B}^k) \leqslant \operatorname{rank}(\tilde{B}^{\hat{k}}) + \operatorname{nz}(\tilde{D}^{\hat{k}}\tilde{D}^k).$$

As a consequence of Proposition 3.2, if $\operatorname{nz}(\tilde{D}^k)$ is larger than $\operatorname{rank}(\tilde{B}^{\hat{k}}) + \operatorname{nz}(\tilde{D}^{\hat{k}}\tilde{D}^k)$, then there is no need to perform computations in GTVERT.

MODIFICATION 2. Step 1 of procedure GTVERT determines the set of positive variables at v by using a threshold value δ^k. If the starting value of δ^k is large, then it must be reduced several times before it is smaller than the value of smallest positive variable at v. On the other hand, if δ^k is very small from the beginning, then the main algorithm must find more accurate solutions before v is identified. Therefore, choosing δ^k too low from the beginning may also increase the computational effort.

We propose a modification to GTVERT to partially alleviate this difficulty. In this modification, in addition to δ^k, we propose the use of an indicator to partition zero and nonzero variables. As a consequence we can start with small value of δ^k with little danger of increased computational effort. This modification is motivated by the following two observations on the behavior of the primal-dual methods.

(i) If $\{x^k\} \to v$ is generated in a primal-dual method, then $(x_i^k - x_i^{k+1})/x_i^k$ computed near v serves as a good indicator for positive and zero variables in v. This is because in the limit implementations show a linear rate of convergence with a small error constant (or possibly superlinear convergence). If

$$\frac{x_i^k - x_i^{k+1}}{x_i^k} \leqslant \epsilon_{\text{rate}} \tag{3.1}$$

for some small $\epsilon_{\text{rate}} < 1$ (say 0.1), then the x_i are expected to be positive at the vertex; otherwise they are assumed zero. Note that the use of (3.1) as an indicator requires minimal computational effort.

(ii) If $\{x^k\} \to v$, then elements on the diagonal of the Cholesky factor L^k computed at each iteration of an interior point method show a clear partition into small and large numbers (pivots). The number of large pivots equals the number of variables positive at v. Small pivots converge to zero as $\{x^k\} \to v$. Without loss of generality assume that the diagonal elements in L^k are

TABLE 3

A MODIFICATION OF THE BASIC PROCEDURE GTVERT

Procedure MGTVERT

Input: x^k, A, δ^k, L^k, ϵ_{rate}, ϵ_{sep}.

Step 1. **Let** $\hat{B}^k = A\hat{D}^k$, where the elements of diagonal matrix $\hat{D}^k$ are given by

$$\hat{D}_{ii}^k = \begin{cases} x_i^k & \text{if } \dfrac{x_i^{k-1} - x_i^k}{x_i^{k-1}} < \epsilon_{\text{rate}}, \\ 0 & \text{otherwise.} \end{cases}$$

Step 2. **Sort** diagonal elements of L^k in decreasing order. **Let** $l = \text{nz}(\hat{D}^k)$. If

$$L_{l+1,l+1}^k / L_{l,l}^k \leq \epsilon_{\text{sep}} \tag{3.2}$$

then **goto** step 3; otherwise **goto** step 4.

Step 3. Same as step 2 in GTVERT with $\hat{D}^k$ and $\hat{B}^k$ replacing $\tilde{D}^k$ and $\tilde{B}^k$. **If** a vertex solution is found then **Exit**; otherwise **goto** step 4.

Step 4. Same as step 1 in GTVERT.

Step 5. Same as step 2 in GTVERT.

permuted in decreasing order and let $l = \text{nz}(v)$. Then as $\{x^k\} \to v$ the ratio $L_{l+1,l+1}^k / L_{l,l}^k$ becomes small.

The indicator (3.1) is different from the indicator recently proposed by Tapia and Zhang [18]. The indicator proposed in [18] is based on the diagonal elements of $D^k A^T (AD^{k^2} A^T)^{-1} AD^k$. Here D^k is the scaling matrix in the interior point methods. The Tapia-Zhang indicator is expensive to use because the inverse of $AD^{k^2}A^T$ is not computed explicitly in practice on large sparse problems. In the context of algorithms for constrained nonlinear programming problems, use of (3.1) as an indicator has been proposed in Tapia [17]. We emphasize that in the current context the use of (3.1) becomes considerably more robust and efficient when we couple it with observation (ii).

The modification of GTVERT which makes use of observations (i) and (ii) is given in Table 3. MGTVERT includes GTVERT to ensure that the procedure to identify vertex solutions remains valid even in case (i) and (ii) fail to hold. Because of this, MGTVERT may compute a Cholesky factor twice. Computational results, however, indicate that the amount of work in GTVERT and MGTVERT is practically the same. Use of MGTVERT often saves a few iterations of the main algorithm in identifying the vertex solution. We recommend its use over GTVERT.

4. A PRIMAL-DUAL PREDICTOR-CORRECTOR METHOD

This section outlines a primal-dual method which we used to perform computational testing. There are several ways to motivate the basic algorithm. The motivation we give here introduces the concept of "prediction-correction" in developing interior point algorithms. The form in which the method is implemented here could also be motivated by considering the approximation of a primal-dual trajectory with a second order Taylor polynomial. A discussion of this can be found in Mehrotra [12], which also discusses many other issues involved in the implementation of interior point methods.

The central idea behind a predictor-corrector approach is as follows. At the beginning of each iteration we have an iterate $x^k > 0$, π^k, $s^k > 0$ in the primal and dual spaces. In our terminology a predictor method is one which only uses the information at x^k, π^k, s^k, and all the previous iterates to generate the new iterate $x^{k+1}, \pi^{k+1}, s^{k+1}$. A predictor-corrector method uses the information used in a predictor method to generate intermediate point(s), and then it makes a sequence of corrections to these points. The correction process is terminated at some stage, and the new iterate is generated.

The method described below generates only one intermediate point and makes one correction to it. The generalization of this method to one which makes more than one correction in a stable way will be discussed in a future paper.

x^*, π^*, s^* is an optimal solution of $P(0)$ if it satisfies

$$\begin{aligned} Ax &= b, \\ A^T\pi + s &= c, \\ XSe &= 0, \\ x \geqslant 0, \qquad s &\geqslant 0. \end{aligned}$$

Given x^k, π^k, s^k, ideally we would like to find $\Delta x, \Delta\pi, \Delta s$ such that

$$\begin{aligned} A(x^k + \Delta x) &= b, \\ A^T(\pi^k + \Delta\pi) + (s^k + \Delta s) &= c, \\ (X^k + \Delta X)(S^k + \Delta S)e &= 0, \\ x^k + \Delta x \geqslant 0, \qquad s^k + \Delta s &\geqslant 0. \end{aligned} \tag{4.1}$$

A rearrangement of terms in (4.1) yields

$$\begin{aligned} A(x^k + \Delta x) &= b, \\ A^T(\pi^k + \Delta\pi) + (s^k + \Delta s) &= c, \\ X^k \Delta s + S^k \Delta x &= -X^k S^k e - \Delta X \Delta S e, \\ x^k + \Delta x \geqslant 0, \qquad s^k + \Delta s &\geqslant 0. \end{aligned} \tag{4.2}$$

We set $\Delta X \Delta S e = 0$ in the right hand side of (4.2) and solve it for Δx, $\Delta\pi$, and Δs while ignoring the nonnegativity constraints. Let $\Delta^1 x, \Delta^1\pi, \Delta^1 s$ represent this solution. $\Delta^1 x, \Delta^1\pi, \Delta^1 s$ computed in this way give the primal-dual affine scaling direction. We use this direction in two ways: (1) to estimate a parameter for a centering direction (centering parameter μ), and (2) to compute an improved (corrected) direction. The use of a centering direction improves the global convergence properties of the method. The correction to the affine scaling direction effectively reduces the number of iterations required to solve the problem.

The centering parameter is estimated as follows:

$$\begin{aligned} \alpha_x^1 &= \min\left(1, \min\left\{ \frac{x_i}{(\Delta^1 x)_i} \,\middle|\, (\Delta^1 x)_i > 0 \right\}\right), \\ \alpha_s^1 &= \min\left(1, \min\left\{ \frac{s_i}{(\Delta^1 s)_i} \,\middle|\, (\Delta^1 s)_i > 0 \right\}\right), \\ \mu &= \left(\frac{(x^k - \alpha_x^1 \Delta^1 x)^T (s^k - \alpha_s^1 \Delta^1 s)}{(x^k)^T s^k} \right)^3 \frac{(x^k)^T s^k}{n}. \end{aligned} \tag{4.3}$$

The motivation for using (4.3) to estimate the centering parameter and its use are discussed in Mehrotra [12].

The corrected direction is now obtained by solving

$$\begin{aligned} A(x^k + \Delta x) &= b \\ A^T(\pi^k + \Delta\pi) + (s^k + \Delta s) &= c \\ X^k \Delta s + S^k \Delta x &= -X^k S^k e - \Delta^1 X \Delta^1 S e + \mu e \end{aligned} \tag{4.4}$$

The solution of (4.4), $\Delta^2 x, \Delta^2 \pi, \Delta^2 s$, is used to obtain the new iterate as follows:

$$\alpha_x^2 = \min\left(1, 0.9995 \min\left\{ \frac{x_i}{(\Delta^2 x)_i} \middle| (\Delta^2 x)_i > 0 \right\}\right),$$

$$\alpha_s^2 = \min\left(1, 0.9995 \min\left\{ \frac{s_i}{(\Delta^2 s)_i} \middle| (\Delta^2 s)_i > 0 \right\}\right),$$

$$x^{k+1} = x^k - \alpha_x^2 \Delta^2 x,$$

$$s^{k+1} = s^k - \alpha_s^2 \Delta^2 s,$$

$$\pi^{k+1} = \pi^k - \alpha_s^2 \Delta^2 \pi.$$

We do not allow step sizes larger than one, because at that value the solutions become feasible.

The directions $\Delta^1 x, \Delta^1 \pi, \Delta^1 s$ and $\Delta^2 x, \Delta^2 \pi, \Delta^2 s$ are computed as follows:

$$D^{k^2} = X^k (S^k)^{-1},$$

$$\Delta^1 \pi = (AD^{k^2} A^T)^{-1} \left[Ax^k - \xi_x^k + A(S^k)^{-1} \xi_s^k \right],$$

$$\Delta^1 s = \xi_s^k + A^T \Delta^1 \pi,$$

$$\Delta^1 x = x^k - D^{k^2} \Delta^1 s,$$

$$\Delta^2 \pi = \Delta^1 \pi + (AD^{k^2} A^T)^{-1} (S^k)^{-1} (\Delta^1 X \Delta^1 S e - \mu e),$$

$$\Delta^2 s = \xi_s^k + A^T \Delta^{k^2} \pi,$$

$$\Delta^2 x = (S^k)^{-1} (-X^k S^k e - \Delta^1 X \Delta^1 S e + \mu e) - D^2 \Delta^2 s,$$

where $\xi_x^k = Ax^k - b$ and $\xi_s^k = A^T \pi^k + s^k - c$.

At each iteration a Cholesky factor is computed once, and it is used two times (i.e., two forward and backward solves).

A starting point for the method was obtained as follows. Let $\tilde{x} = A^T (AA^T)^{-1} b$, find $\text{xsft} = \max(-2 \min\{\tilde{x}_i\}, 10^{-2} \|b\|)$ and let $x_i^0 = \tilde{x}_i + \text{xsft}$

for $i = 1, 2, \ldots, n$. For dual starting point, let $\pi^0 = 0$, find ssft $= \max(-2\min\{c_i\}, 0) + 1$, and let $s_i^0 = c_i + \text{ssft}$ for $i = 1, 2, \ldots, n$.

A comparison with the results in Choi, Monma, and Shanno [3] and Lustig, Marsten, and Shanno [7] shows that the implemented method requires considerably fewer (by up to 50%) iterations to achieve the same accuracy in the objective value as in [3, 7]. The number of iterations is also smaller than required to solve the problems in Boggs, Domich, Donaldson, and Witzgall [2]. The implementation in [2] computes three directions and solves a linear program in a lower dimensional subspace at each iteration.

5. COMPUTATIONAL RESULTS

The test problems used for the computational results given in this section were obtained from the NETLIB [4] test set. Problems with no bounds are solved. The test set has a mix of (primal and dual) degenerate problems, which vary in the amount of degeneracy. The statistics for these problems are not given here, since they are easily available from NETLIB.

All problems were processed to remove easily identifiable (primal) variables with fixed value over the feasible set by using the procedure outlined in Mehrotra [11]. The fixed variables are easily restored to find a vertex solution of the original problem, once a vertex solution of processed problem is found. The objective value reported in the results is computed at the vertex of the original problem.

The following parameters were used:

$$\delta^0 = 10^{-6}, \quad \delta^{k+1} \leftarrow \delta^k \qquad (\text{in GTVERT and MGTVERT}),$$

$$\epsilon_{\text{sep}} = 10^{-6}, \quad \epsilon_{\text{rate}} = 10^{-1} \qquad (\text{in MGTVERT}),$$

$$\epsilon = 10^{-6} \min\left(|\underline{z}^k|, |c^T v^*(r)|\right) \qquad (\text{in ENCVERT}).$$

$\underline{z}^k$ was computed as in (2.8) using the solutions at the iteration after which the cost vector was perturbed. $M \approx \sum_{i=1}^{n} x_i^k$ was used in computing $\underline{z}^k$. Hence, by the stopping criterion approximately six digits of accuracy was ensured in the objective value at the vertex solution. $\underline{z}^k$ was used in place of z^* to verify that the vertex solution obtained at termination satisfied the stopping criterion. The cost vector was perturbed, i.e., step 1 in ENCVERT was terminated, after $|\beta^k| \leqslant 10^{-7}\underline{z}^k$ was satisfied. For all problems it was enough to solve $P(r)$ once.

Table 4 gives the computational results obtained on the problems in the test set. Column 2 of this table gives the number of iterations after which $P(r)$ was introduced. Column 3 gives the number of iterations on $P(r)$ after which a vertex solution was found, while using procedure MGTVERT. Modification 1 was incorporated while implementing MGTVERT. In our computational experience the performance of GTVERT and MGTVERT was similar, MGTVERT being superior for some problems. Since we recommend MGTVERT, we only include results obtained from its use.

Column 5 gives the number of times a Cholesky factor was computed in MGTVERT. Columns 6 and 7 are intended to give an idea of the amount of primal degeneracy present in the problems. Column 6 gives $\text{nz}(\tilde{D}^k)$ at the first call to MGTVERT. MGTVERT was called for the first time at the iteration at which $P(r)$ was generated. Column 7 gives $\text{nz}(v)$ at the vertex solution obtained at termination. Column 8 gives the objective value at the vertex solution, and Column 9 gives the objective value given in the *problem index file* available from NETLIB.

We make following general remarks on the performance of the proposed approach.

(1) The proposed approach is highly reliable. One perturbation was sufficient for all the problems to force the solution to go to a vertex. Although care is needed when taking decisions based on floating point computations (to guard against numerical errors), the results show that it is possible to develop robust and efficient procedures.

(2) Two types of behavior were seen when analyzing the number of iterations performed in solving $P(r)$. In problems for which the solutions of $P(0)$ were converging to a vertex, its identification was almost immediate. In problems for which the solutions were in the interior of an optimal face, it took on the average 40% additional iterations before a vertex solution was identified. This shows that on the average the additional work performed to identify the vertex solution is comparable to that required to find a point near the optimal face. It may be possible to significantly reduce this work by further processing $P(r)$. This remains a topic for future work. Perhaps what is more significant in evaluating the merits of the proposed approach is the observation from the results that the number of additional iterations required to find the vertex solution is independent of (or very weakly dependent on) the amount of degeneracy present in the degenerate problems.

(3) The results in column 4 show that the Cholesky factor is computed only a small number of times in MGTVERT. This shows that Modification 1 proposed in Section 3.2 is very effective.

(4) It is also interesting to observe that for all the problems (except `stocfor1`) the accuracy in the objective value at the vertex solution was considerably better than expected from the bound in Proposition 2.2.

TABLE 4

COMPUTATIONAL RESULTS ON THE VERTEX FINDING PROBLEM

Problem	it $P(0)$	MGT	$\tilde{L}$	nz(D^k)	nz(v)	$z(v^*(r))$	z(NETLIB)
afiro	7	6	2	22	20	−4.6475314285714E+2	−4.6475314285714E+2
adlittle	12	7	1	71	55	2.2549496316238E+5	2.2549496316238E+5
scagr7	12	0	1	128	128	−2.3313898243310E+6	−2.3313892547843E+6
stocfor1	17	1	2	101	96	−4.1131976219434E+4	−4.1131983232E+4
sc205	10	0	1	191	191	−5.2202061211715E+1	−5.2202061211707E+1
share2b	10	5	2	105	86	−4.1573224074172E+2	−4.1573224074142E+2
share1B	21	0	1	112	112	−7.6589318579050E+4	−7.6589318579186E+4
scorpion	14	0	1	259	259	1.8781248227381E+3	1.8781248227381E+3
scagr25	15	0	1	426	426	−1.4753433060769E+7	−1.4753433060769E+7
sctap1	15	8	2	337	220	1.4122500000000E+3	1.4122500000000E+3
brandy	17	7	2	132	127	1.5185098964881E+3	1.5185098964881E+3
scsd1	9	4	2	31	11	8.6666666743334E+0	8.6666666743334E+0
israel	30	10	1	193	174	−8.9664482186252E+5	−8.9664482186305E+5
bandm	16	0	1	244	244	−1.5862801845012E+2	−1.5862801845012E+2
scfxm1	16	7	2	300	284	1.8416759028349E+4	1.8416759028349E+4
e226	21	9	2	197	192	−1.8751929066370E+1	−1.8751929066371E+1
agg	22	8	2	394	388	−3.5991767286581E+7	−3.5991767287E+7
scrs8	19	7	2	320	299	9.0429695380079E+2	9.0429695380079E+2

Problem	it $P(0)$	MGT	$\bar{L}$	nz(D^k)	nz(v)	$z(v^*(r))$	z(NETLIB)
beaconfd	9	7	2	118	114	3.3592485807200E+4	3.3592485807200E+4
scsd6	10	5	2	182	63	−5.0500000078262E+1	−5.0500000078262E+1
ship04s	13	2	2	218	217	1.7987147004454E+6	1.7987147004454E+6
agg2	18	9	1	549	509	−2.0239252355977E+7	−2.0239252356E+7
agg3	17	7	2	549	511	1.0312115935089E+7	1.0312115935E+7
scfxm2	18	9	3	649	584	3.6660261564999E+4	3.6660261564999E+4
ship041	12	2	2	275	274	1.7933245379704E+6	1.7933245379704E+6
fffff800	32	7	3	463	374	5.5567956481750E+5	5.5567996085451E+5
ship08s	14	0	1	260	260	1.9200982105346E+6	1.9200982105346E+6
sctap2	12	8	3	1019	681	1.7248071428571E+3	1.7248071428571E+3
scfxm3	19	9	2	930	884	5.4901254549751E+4	5.4901254549751E+4
ship12s	17	1	2	342	339	1.4892361344061E+6	1.4892361344061E+6
scsd8	10	12	2	551	245	9.0499999992544E+2	9.0499999992546E+2
stocfor2	22	0	1	1809	1809	−3.9024408537891E+4	−3.9024408538E+4
sctap3	13	12	3	1276	858	1.4240000000000E+3	1.4240000000000E+3
czprob	34	7	1	748	688	2.1851966988566E+6	2.1851966988566E+6
25fv47	25	9	3	743	725	5.5018458882866E+3	5.5018458882867E+3
ship081	16	0	1	429	429	1.9090552113891E+6	1.9090552113891E+6
ship121	19	3	2	588	587	1.4701879193293E+6	1.4701879193293E+6

I thank two anonymous referees for many useful suggestions which led to improvements in the presentation of this paper.

REFERENCES

1 I. Adler, N. Karmarkar, M. G. C. Resende, and G. Veiga, An implementation of Karmarkar's algorithm for linear programming, *Math. Programming* 44(3): 297–336 (1989).

2 P. T. Bogs, P. D. Domich, J. R. Donaldson, and C. Witzgall, Optimal 3-Dimensional Methods for Linear Programming, NISTIR 89-4225, Center for Computing and Applied Mathematics, U.S. Dept. of Commerce, National Inst. of Standards and Technology, Gaithersburg, MD 20899, 1989.

3 I. C. Choi, C. L. Monma, and D. F. Shanno, Further Development of a Primal-Dual Interior Point Method, Technical Report, Rutgers Center for Operations Research, Rutgers Univ., New Brunswick, NJ 08903, 1989.

4 D. M. Gay, Electronic mail distribution of linear programming test problems, *Math. Programming Soc. Committee on Algorithms News Letter*, Dec. 1985, pp. 10–12.

5 P. E. Gill, W. Murray, M. A. Saunders, J. A. Tomlin, and M. H. Wright, On projected Newton barrier methods for linear programming and an equivalence to Karmarkar's projective method, *Math. Programming* 36:183–209 (1986).

6 D. Goldfarb and S. Mehrotra, A self-correcting version of Karmarkar's algorithm, *SIAM J. Numer. Anal.* 26, No. 4 (1989).

7 I. J. Lustig, R. E. Marsten, and D. F. Shanno, Computational Experience with a Primal-Dual Interior Point Method for Linear Programming, TR J-89-11, Industrial and Systems Engineering Report Series, Georgia Inst. of Technology, Atlanta, GA 30332, Oct. 1989.

8 R. E. Marsten, M. J. Saltzman, D. F. Shanno, G. S. Pierce, and J. F. Ballintijn, Implementation of a dual affine interior point algorithm for linear programming, *ORSA J. Comput.* 1(4):287–297 (1989).

9 K. A. McShane, C. L. Monma, and D. Shanno, An implementation of a primal-dual interior point method for linear programming, *ORSA J. Comput.* 1(2):70–83 (1989).

10 S. Mehrotra, Implementations of Affine Scaling Methods: Approximate Solutions of Systems of Linear Equations Using Preconditioned Conjugate Gradient Methods, TR89-04R, Dept. of IE/MS, Northwestern Univ., Evanston, IL 60208, 1989.

11 S. Mehrotra, Implementations of Affine Scaling Methods: Towards Faster Implementations with Complete Cholesky Factor in Use, TR89-15, Dept. of IE/MS, Northwestern Univ., Evanston, IL 60208, 1989.

12 S. Mehrotra, On the Implementation of a (Primal-Dual) Interior Point Method, TR90-03, Dept. of IE/MS, Northwestern Univ., Evanston, IL 60208, 1990.

13 S. Mehrotra, On an implementation of primal-dual predictor-corrector algorithms, presented at Second Asilomar Workshop on Progress in Mathematical Programming, Asilomar, Calif., 1990.

14 C. L. Monma and A. J. Morton, Computational experience with a dual affine variant of Karmarkar's method for linear programming, *Oper. Res. Lett.* 6(6):261–267 (1987).

15 G. L. Nemhauser, A. H. G. Rinnooy Kan, and M. J. Todd (Eds.), *Handbook in Operations Research and Management Sciences, Vol. 1, Optimization*, 1989.

16 C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization Algorithms and Complexity*, Prentice-Hall, 1982.

17 R. A. Tapia, On the role of slack variables in quasi-Newton methods for constrained optimization, in *Numerical Optimization of Dynamic Systems* (L. C. W. Dixon and G. P. Szego, (Eds.), North-Holland, 1980.

18 R. A. Tapia and Y. Zhang, A Fast Optimal Basis Identification Technique for Interior Point Linear Programming Methods, TR89-1, Dept. of Mathematical Sciences, Rice Univ., Houston, TX 77251, rev. Oct. 1989.

Received 9 February 1990; final manuscript accepted 8 November 1990