Sphere Methods for LP

Katta G. Murty
Department of Industrial and Operations Engineering,
University of Michigan,
Ann Arbor, MI 48109-2117, USA
Phone: 734-763-3513,
Fax: 734-764-3451
murty@umich.edu
www-personal.engin.umich.edu/~ murty

Mohammad R. Oskoorouchi
Department of Information Systems and Operations Management,
College of Business Administration,
California State University San Marcos,
San Marcos, CA 92096-0001, USA,
moskooro@csusm.edu
www.csusm.edu/oskoorouchi

December 2008

Abstract

(Murty and Oskoorouchi [6]) discussed a new sphere method for solving LPs based on earlier work of (Murty [7, 8]), and showed computational results on it which are very encouraging. In this paper we discuss some enhancements to that method and show that they improve its performance. We also discuss an improved version of that method, Sphere method-2 which improves the performance by 20% to 40%.

Key words: Linear Programming (LP), Interior point methods (IPM.s) , ball centers of a polytope, solving LPs using matrix inversions sparingly, Sphere methods-1, 2 for LPs.

1 Introduction

Among mathematical models, Linear Programming (LP) is the most commonly used in decision making applications. The simplex method (Dantzig and Thappa [1]) developed in mid-20th century, and Interior Point Methods (IPMs) developed in the last quarter of the 20th century (among them in particular the primal-dual path following IPMs, [2 to 5, 10 to 13]) are currently the commonly used algorithms for solving LP models in software systems. These software implementations are able to solve large scale models (those involving thousands of constraints) within reasonable times, which has made them very popular in practice.

While solving large scale LP models, typically the simplex method takes many steps, however each of these involves much less work than a step in an IPM. IPMs have been observed to take much smaller number of steps, and this number grows very slowly with the size of the LP model being solved; with the result that IPMs have gained the reputation of taking almost a constant number of steps even as the size of the LP grows.

Both these existing classes of algorithms for LP are based on full matrix inversion operations, with every constraint in the model appearing in the computational process in every step. In large scale applications these matrix inversion operations limit the ability of these algorithms to only those in which the coefficient matrix is very sparse. As the density of the coefficient matrix increases, typically the effectiveness of these algorithms fades.

LP models which do not have the property of being very sparse do arise in many application areas, and in some areas the models may be 100% dense. The sphere method for LP developed in [6] is an IPM that can handle all such models along with the others, without any problems, because it uses matrix inversion operations very sparingly. In any step of the sphere method only a small subset of constraints (called the **touching set of constraints**) appear in the matrix inversion operations, And redundant constraints, if any in the original model, are automatically guaranteed never to appear in the touching set. And in our computational experiments, we observed that the number of iterations taken by the sphere method to solve large scale models is also very small, and grows very slowly with the size of the model, like in other IPMs.

The sphere method needs an initial interior feasible solution. Each iteration of the method begins with the best interior feasible solution obtained at the end of the previous iteration; and consists of two steps: a **centering step**, and a **descent step**.

In concept, the aim of the centering step is to find a **ball center**, which is an interior feasible solution with objective value equal to or better than that of the current interior feasible solution, and is the center of a largest radius ball inside the feasible region of the original LP subject to the constraint on its center. This centering step takes up most of the computational effort in the iteration. Once the ball center is obtained, the descent step which is computationally cheap carries out several descent steps from it, and the iteration stops with the best point obtained in all these descent steps.

The sphere method [6] considers LPs in the form:

$$\begin{aligned} \text{Minimize} \quad & z = cx \\ \text{subject to} \quad & Ax \geq b \end{aligned} \tag{1}$$

where A is an $m \times n$ data matrix; with a known interior feasible solution x^0 (i.e., satisfying $Ax^0 > b$). Strategies for modifying any given LP into this form are discussed in [6]. Let K denote its set of feasible solutions, and K^0 its interior. We assume that c, and each row vector of A is normalized so that $||c|| = ||A_{i.}|| = 1$ for all $i = 1$ to m. In [6] the following concepts used in the sphere method are defined.

Largest inscribed ball $B(x, \delta(x))$ inside K with x as center, for $x \in K^0$: It is the largest ball with x as center that can be inscribed in K, and $\delta(x) = \min\{A_{i.}x - b_i : i = 1 \text{ to } m\}$ is its radius. So, $B(x, \delta(x)) = \{y : ||y - x|| \leq \delta(x)\}$.

A ball center of K: It is a point $x \in K^0$ such that $B(x, \delta(x))$ is a largest ball that can be inscribed in K, i.e., x maximizes $\delta(y)$ over $y \in K^0$.

A ball center of K on the objective plane $H = \{x : cx = t\}$: It is a point $x \in H \cap K$ that maximizes $\delta(y)$ over $y \in H \cap K$.

The index set of touching constraints in (1), $T(x)$: Defined for $x \in K^0$, it is the set of all indices i satisfying: $A_{i.}x - b_i = \text{Minimum}\{A_{p.}x - b_p : p = 1 \text{ to } m\} = \delta(x)$. The facetal hyperplane $\{x : A_{i.}x = b_i\}$ is a tangent plane to $B(x, \delta(x))$ for each $i \in T(x)$.

GPTC (gradient projection on touching constraint) directions: Let c^i denote the orthogonal projection of c^T on $\{x : A_{i.}x = 0\}$, i.e., $c^i = (I - A_{i.}(A_{i.})^T)c^T$ for $i = 1$ to m. When the ball $B(x, \delta(x))$ is under consideration, the diections $-c^i$ for $i \in T(x)$ are called the GPTC directions at the current center x.

In practice, a ball center of K may not be unique; and in case it is not unique, [6] discusses a conceptual definition of identifying a specific one among them to be called "**the ball center of** K". This theoretical definition guarantees that for every polytope the ball center is well defined and unique; and correspondingly the ball center of K on the objective plane $H = \{x : cx = t\}$ is well defined and unique for all values of t satisfying $H \cap K^0 \neq \emptyset$.

[6] also discusses techniques for computing a ball center of K, or a ball center of K on a given objective plane H, approximately, using a series of line search steps. In each of these steps, at the current point $\bar{x}$, the algorithm in [6] selects a direction y which is a **profitable direction** to move at $\bar{x}$, i.e., $\delta(\bar{x} + \alpha y)$ strictly increases as α increases from 0; and determines the optimum step length to maximize $\delta(\bar{x} + \alpha y)$ over $\alpha \geq 0$ (this optimum step length is obtained by solving a 2-variable LP).

A direction y has been shown in [6] to be a profitable direction at $\bar{x} \in K^0$ iff $A_{i.}y > 0$ for all $i \in T(\bar{x})$, so it easy to check whether any given direction y is a profitable direction at the current point.

Once a profitable direction y at the current point $\bar{x}$ has been determined, the optimum step length α in this direction that maximizes $\delta(\bar{x} + \alpha y)$ over $\alpha \geq 0$ is $\bar{\alpha}$, where $(\bar{\delta}, \bar{\alpha})$ is the optimum solution of the following 2-variable LP.

$$\begin{array}{rlll} & \text{Maximize} \quad \delta & & \\ \text{subject to} & \delta - \alpha A_{i.}y & \leq & A_{i.}\bar{x} - b_i \qquad i = 1, \ldots, m \\ & \delta, \alpha \geq 0 & & \end{array} \tag{2}$$

and $\bar{\delta}$ is the optimum objective value $\delta(\bar{x} + \bar{\alpha}y)$. So, the line search for the maximum value of δ in the direction y involves solving this 2-variable LP, which can be carried out efficiently (e.g., by the simplex algorithm).

Two procedures for generating profitable directions are discussed in [6], one is LSFN which selects a direction among those in $\Gamma_1 = \{\pm A_{i.}^T : i = 1 \text{ to } m\}$. The other is LSCPD which obtains profitable directions by solving a system of linear equations. For computing a ball center of K on the objective plane through the current point $\bar{x}$, only profitable directions y satisfying $cy = 0$ are considered.

2 Sphere Method 1

We will call the method discussed in [6] as "**Sphere method 1**". In concept, the centering step in this method has the aim of finding a ball center of K on the objective plane through the current point. So, the LSFN sequence of steps in it use profitable directions from the set $\Gamma_2 = \{\pm P_{.i} : i = 1 \text{ to } m\}$, where $P_{.i}$ is the orthogonal projection of $A_{i.}^T$ on $\{y : cy = 0\}$. But the LSCPD steps in it generate and use profitable directions y which satisfy $cy \leq 0$, so some of them may also decrease the objective value cy. Here is a summary of the centering step in Sphere Method 1 in general iteration $r + 1$.

The centering Step in Iteration $r+1$ in Sphere method 1: Let x^r be the initial interior feasible solution for this iteration. This step consists of a series of line searches in profitable directions with the aim of finding an x that maximizes $\delta(x)$ subject to the constraint $cx \leq cx^r$. In each of these line searches, given the search direction, the optimum step length to take in that direction is determined by solving a 2-variable LP of the form (2) as described above. First it carries out the LSFN sequence of line searches selecting profitable search directions from the set of facetal normal directions (the name LSFN comes from this). After the LSFN sequence, it carries out a LSCPD sequence of line searches using computed profitable directions (the name LSCPD comes from this). We describe each of these sequences briefly.

The LSFN sequence of line searches: Beginning with the initial point x^r, this generates a sequence of points $x^{r,k}$, $k = 1, 2, \ldots$ along which the radius of the ball δ is strictly increasing.

At the current point $x^{r,k}$, it selects a profitable directions from the set $\Gamma_2 = \{\pm P_{.1}, ..., \pm P_{.m}\}$, where $P_{.i} = (I - c^T c)A_{i.}^T$, the orthogonal projection of $A_{i.}^T$ (the direction normal to the facet of K defined by the i-th constraint in (1)) on the hyperplane $\{x : cx = 0\}$, for $i = 1$ to m, instead of the set Γ_1 as discussed above. So any step length from a point in the current objective plane, in a direction from Γ_2, will keep the point on the current objective plane. The procedure continues as long as profitable directions for line search are found in Γ_2, and this sequence terminates with the final point which we denote by $\tilde{x}^r$.

The LSCPD sequence of line searches: Beginning with the initial point $\tilde{x}^r$ obtained at the end of the LSFN sequence, this generates a sequence of points $\tilde{x}^{r,k}$, $k = 1, 2, ...$ along which the radius of the ball δ is strictly increasing.

When $\tilde{x}^{r,k}$ is the current solution, it selects the profitable direction to move to be a solution of the system of linear equations:

$$A_{i.}y = 1 \quad \text{for all } i \in T(\tilde{x}^{r,k}). \tag{3}$$

satisfying $cy \leq 0$. If a solution to this system satisfying $cy \leq 0$ is obtained (for details see Section 4.1 in [6]), the line search is carried out with that solution as the search direction, and determining the optimum step length to move as in (2); and the procedure continues the same way with the point obtained at the end of this line search. It can be verified that the index set of touching constraints $T(\tilde{x}^{r,k})$ grows with k, so the system of linear equations (3) grows by at least one more constraint after each line search in this sequence. The sequence stops when either (3) has no solution satisfying $cy \leq 0$, or when the set of coefficient vectors of the constraints in it becomes linearly dependent. That's why the number of steps in this sequence is at most n.

It is shown in [6] that this entire sequence needs a single matrix inversion, carried out in stages adding one row and column to the basis matrix at a time. Formulas for efficiently updating the basis inverse for (3) along this sequence are given in [6]. Let $\bar{x}^r$ denote the final point obtained at the end of the LSCPD sequence, it is the approximate ball center obtained in this centering step, the iteration moves to the descent step with it.

The descent step in this iteration actually carries out several descent steps labeled D1, D2, D3, D4, D5.1, and selects the best point obtained from all of them as the output of this

iteration. Here is a summary of all the descent steps in each iteration of Sphere Method 1 in general iteration $r+1$.

Descent Steps in Iteration $r+1$ in Sphere Method 1: Let $\bar{x}^r$ denote the approximate ball center obtained in the centering step of this iteration.

Each descent step carried out in this iteration requires one minimum ratio computation. For example, consider a descent step from the current center $\bar{x}^r$ in the descent direction y (i.e., satisfying $cy < 0$). If the step length is λ, the move leads to the point $\bar{x}^r + \lambda y$. Select a small positive number ϵ_1 as the tolerance for minimum $\{A_{i.}x - b_i : i = 1 \text{ to } m\}$ for the point x to be in the interior of K. Then we will take the step length from $\bar{x}^r$ in the direction y to be: $(-\epsilon_1)$ + (the maximum step length possible while remaining inside K), which is

$$\gamma = \text{minimum}\{\tfrac{-A_{i.}\bar{x}^r + b_i + \epsilon_1}{A_{i.}y} : i \text{ such that } A_{i.}y < 0\}$$

and then the point obtained at the end of this descent step will be $\bar{x}^r + \gamma y$ if γ is finite.

If $\gamma = \infty$, the objective function $z(x)$ is unbounded below in (1), and $\{\bar{x}^r + \lambda y : \lambda \geq 0\}$ is a feasible half-line along which $z(x)$ diverges to $-\infty$ on K. Terminate the method if this occurs.

We now list the various descent steps carried out in this iteration. After each descent step, include the point obtained at the end of it, along with its objective value, in a **List**.

D1, Descent Step 1: From the ball center $\bar{x}^r$ take a descent step in the direction $d^1 = -c^T$.

D2, Descent Step 2: From the ball center $\bar{x}^r$ take a descent step in the direction $d^2 = \bar{x}^r - \bar{x}^{r-1}$, where $\bar{x}^{r-1}$ denotes the ball center computed in the previous iteration r. So, this direction is the direction of the path of ball centers generated in the algorithm.

D3, Descent Steps 3: Carry out descent steps from the ball center $\bar{x}^r$ in each of the GPTC directions at $\bar{x}^r$. After these descent steps are carried out, define

d^3 = direction among the GPTC directions that gives maximum reduction in objective value when the descent step is taken from the center $\bar{x}^r$.

D4, Descent Step 4: From the ball center $\bar{x}^r$ take a descent step in the direction $d^4 = (\sum(-c^i : \text{for } i \in T(\bar{x}^r))/|T(\bar{x}^r)|$, the average direction of all the GPTC directions at $\bar{x}^r$.

D5.1, Descent Steps 5.1: For $i \in T(\bar{x}^r)$, let x^{ir} denote the orthogonal projection of the center $\bar{x}^r$ on the touching facetal hyperplane $\{x : A_{i.}x = b_i\}$; it is the point where this facetal hyperplane touches the ball $B(\bar{x}_r, \delta(\bar{x}^r))$. The points x^{ir} for $i \in T(\bar{x}^r)$ are called the **touching points (TPs)** of the ball $B(\bar{x}_r, \delta(\bar{x}^r))$ with its touching facetal hyperplanes of K. See Figure 1

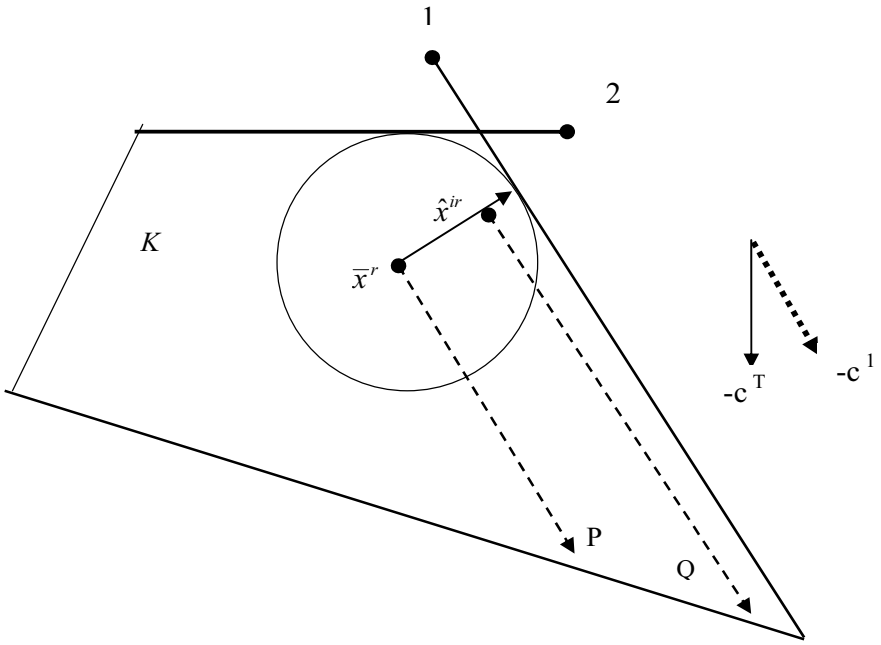

Figure 1: Figure 1: Descent steps in a GPTC direction. Here $\bar{x}^r$ is the current center, $T^(\bar{x}^r) = \{1, 2\}$. Directions $-c^T$ pointing down south, $-c^1$ = orthogonal projection of $-c^T$ on facetal hyperplane of constraint 1, are shown. x^{1r} = TP of constraint 1, $\hat{x}^{1r}$ = NTP corresponding to constraint 1. Descent step from $\bar{x}^r$ $[\hat{x}^{1r}]$ in direction $-c^1$ are shown, leading to points P $[Q]$ respectively. Here Q is a much better point than P.

Let $0 < \epsilon < 1$ be a small positive tolerance ($\epsilon = 0.1$ works well). Then for $i \in T(\bar{x}^r)$, the point on the line segment joining $\bar{x}^r$ and x^{ir} close to the TP x^{ir},

$$\hat{x}^{ir} = \epsilon\bar{x}^r + (1 - \epsilon)x^{ir}$$

is called the **near touching point (NTP)** corresponding to the tangent plane $\{x : A_{i.}x = b_i\}$

of the ball $B(\bar{x}_r, \delta(\bar{x}^r))$.

D5.1 consists of $|T(\bar{x}^r)|$ descent steps: for each $i \in T(\bar{x}^r)$, it carries out a descent step in the GPTC direction $-c^i$ from the NTP $\hat{x}^{ir}$. The output of D5.1, denoted by $\tilde{x}^{r1}$ is the best point obtained in it.

When all these descent steps are carried out, the best point among the output points of all the descent steps is the output of this iteration, with that point the method goes to the next iteration. Just as other IPMs, this method also terminates when the change in the final points obtained in successive iterations is smaller than some tolerance (i.e., it terminates at the end of Iteration $r+1$ if $||x^{r+1} - x^r||/||x^r|| < \epsilon$, concluding that x^{r+1} is an optimum solution of (1)).

In Section 3, we will discuss the improved version called Sphere method 2; and in Section 4 we provide our computational results comparing the two methods.

3 Sphere Method 2

In Sphere method 1 the set of feasible solutions considered remains unchanged (i.e., remains the original K) throughout the algorithm; but the current objective plane $\{x : cx = t\}$ keeps on sliding parallely towards decreasing values of t from one iteration to the next. The centering step in this method in each iteration, has the aim of finding a ball center on the current objective plane, at least in principle. Even though line search directions y used in LSCPD in the Centering Step in Sphere method 1 may satisfy $cy < 0$; all the search directions used in LSFN satisfy $cy = 0$, and hence leave the objective value unchanged.

In Sphere method 2, in contrast, the set of feasible solutions K considered is updated by the current objective value after each iteration, and hence gets smaller. So, to distinguish, we will denote by K^r, the set of feasible solutions considered in Step r, and we will have $K^r \subset K^{r-1} \subset K$ for all r. And in the centering step of Sphere method 2, all line search directions used (in both the LSFN and LSCPD sequences) will both be profitable and strict descent directions for the original objective function $z = cx$.

Also, in [9] two additional descent steps, D5.2, D6 to be used in every iteration of the sphere method have been proposed. Of these D6 performed poorly in comparison with other descent

steps in preliminary steps, hence we will ignore it. But we will include D5.2 as an additional descent step in every iteration of Sphere method 2.

The first iteration of Sphere method 2 begins with the initial interior feasible solution x^0. We will now describe the general iteration, Iteration $r+1$, in this method.

General Iteration $r+1$

The initial point for this iteration is x^r, the interior feasible solution obtained at the end of the previous iteration. Define the set of feasible solutions to be considered for this iteration to be K^{r+1}, where

$$K^{r+1} = \{x : Ax \geq b, \quad \text{and} \quad cx \leq cx^r + \epsilon\}$$

where ϵ is a small positive tolerance parameter. Go to the centering step in this iteration.

Centering Step: The aim of this step is to find a ball center of K^{r+1} approximately, as described earlier (also in Section 2.1 of [6]).

LSFN: The set of facetal normal directions of K^{r+1} is $\Gamma_1^{r+1} = \{\pm c^T, \pm A_{i.}^T : i = 1 \text{ to } m\}$. Apply the LSFN sequence to find a ball center for K^{r+1} as described above using profitable directions for K^{r+1} from Γ_1^{r+1}.

LSCPD: This sequence begins with the interior feasible solution obtained at the end of LSFN.

Let $\hat{x}$ denote the interior feasible solution in a step of this sequence. The touching constraint set at $\hat{x}$ for K^{r+1} will typically include the objective constraint in the definition of K^{r+1}. If it does not, then apply this sequence as discussed earlier (also described in detail in Section 2.1 of [6]).

On the other hand, if the touching constraint set includes the objective constraint, let $T^{r+1}(\hat{x})$ denote the touching constraint index set for K^{r+1}. Solve the system

$$\begin{aligned} A_{i.}y &= 1 \quad \text{for all} \quad i \in T^{r+1}(\hat{x}) \\ -cy &= \beta \end{aligned} \tag{4}$$

where β is a positive parameter. Earlier (and in Section 2.1 of [6]) we used only $\beta = 1$. But here we will leave it as a parameter which is restricted to take positive values only; and obtain a solution of (4) as a function of this parameter β. Let this solution be denoted by $p + \beta q$.

As in Section 2.1 of [6], if B is a basis associated with the basic vector y_B obtained for (4), let y_D denote the vector of remaining nonbasic variables in (4) associated with the basic vector y_B. Let $p = (p_B, p_D),\ q = (q_B, q_D)$ be the partition of the vectors p, q corresponding to the partition of y into basic, nonbasic parts (y_B, y_D). Then $q_D = p_D = 0$, and q_B is the last column of B^{-1}, and p_B is the sum of the remaining columns of B^{-1}.

So, for all $\beta > 0$, $p + \beta q$ is a profitable direction at $\hat{x}$ for K^{r+1}. With $p + \beta q$ as line search direction, the optimum step length α (maximizing $\delta(\hat{x} + \alpha(p + \beta q))$, the radius of the maximum radius ball inscribed in K^{r+1} with $\hat{x} + \alpha(p + \beta q)$ as center) is determined by solving the 3 variable LP in variables δ, α, γ

$$\begin{aligned} \text{Maximize} \quad \delta \qquad & \qquad \text{subject to} \\ \delta - \alpha A_{i.}p - \gamma A_{i.}q \quad &\leq \quad A_{i.}\hat{x} - b_i, \quad i = 1, ..., m \\ \delta - \alpha(-c)p - \gamma(-c)q \quad &\leq \quad (-c)\hat{x} - ((-c)\hat{x} - \epsilon) \\ \delta, \alpha, \gamma \quad &\geq 0. \end{aligned}$$

Here α, γ will both be > 0 at optimum. Actually this γ is $(\alpha)(\beta)$.

If $(\bar{\delta}, \bar{\alpha}, \bar{\gamma})$ is an optimum solution of this 3-variable LP, then the point obtained at the end of this step is $\hat{x} + \bar{\alpha}p + \bar{\gamma}q$. With that the next LSCPD step is applied again as here, and so on until the LSCPD sequence is completed,

Let $\bar{x}$ denote the point obtained at the end of LSCPD, it is the approximate ball center of K^{r+1} obtained in this iteration. See Figure 2.

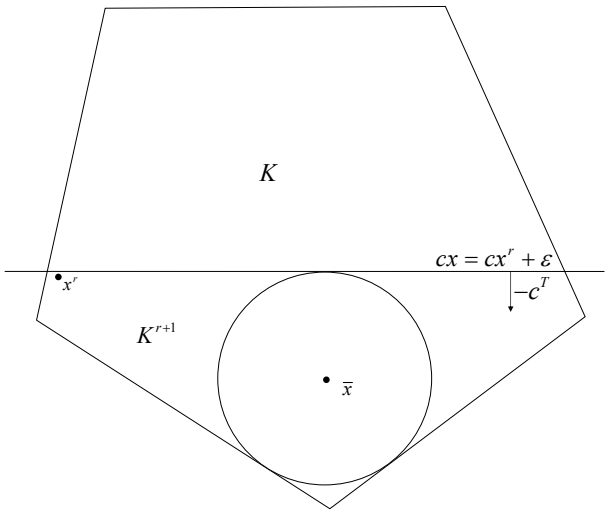

Figure 2: Figure 2: K is the original set of feasible solutions of the LP being solved. The current set of feasible solutions in an iteration when x^r is the initial interior feasible solution, is K^{r+1}. The ball shown is the largest ball inside K^{r+1}, and its center $\bar{x}$ is a ball center obtained in the centering step in this iteration.

The Descent Steps: With the point $\bar{x}$ obtained at the end of the centering step, the iteration moves to the Descent steps in this iteration for the current set of feasible solutions K^{r+1}.

It first applies descent steps D1 to D5.1 as described in Sphere method 1 in the current set of feasible solutions K^{r+1}. Let $\tilde{x}^{r1}$ denote the best point (by objective value) obtained in descent steps D1 to D5.1. This $\tilde{x}^{r1}$ is the initial interior feasible solution for Descent Step 5.2 (D5.2).

D5.2, Descent Step 5.2: By the way the descent steps are carried out, it is clear that $\tilde{x}^{r1}$ is close to the boundary of K^{r+1}, and $\delta(\tilde{x}^{r1}) \leq \epsilon_1$. Find the touching set $T(\tilde{x}^{r1}) =$ set of all constraint indices for the current set of feasible solutions that tie for the minimum in $\{A_{i.}\tilde{x}^{r1} - b_i : i = 1 \text{ to } m;\ -c\tilde{x}^{r1} + cx^r + \epsilon\ \}$.

For each $i \in T(\tilde{x}^{r1})$, from $\tilde{x}^{r1}$ take a descent step in the GPTC direction $-c^i$ and include the resulting point along with its objective value in a new **List 5.2**.

At the end, let $\tilde{x}^{r2}$ denote the best point in List 5.2 by objective value. If $c\tilde{x}^{r1} - c\tilde{x}^{r2}$ is:

$\leq$ some selected tolerance for objective value reduction, take $\tilde{x}^{r2}$ as the output of this Descent Step 5.2, put $\tilde{x}^{r2}$ along with its objective value in the List.

$>$ the selected tolerance for objective value reduction, with $\tilde{x}^{r2}$ as the initial interior feasible solution repeat this Descent Step 5.2; and continue the same way.

When all these descent steps are carried out, the best point among the outputs of all the descent steps carried out in this iteration, is the output of this iteration. With that point the method goes to the next iteration. Termination criteria are the same as in Sphere method 1, as described in [6].

Instead of giving β the specific value 1 as in earlier methods, leaving it as a positive parameter in (2), improves the performance of the centering step in Sphere method 2.

4 Preliminary Computational Results on Sphere Methods 1 and 2

In this section we discuss the results of our computational tests on sphere methods 1 and 2. Broadly these results indicate that Sphere method 2 is up to 40% better than Sphere method 1.

We use MATLAB 7.0 routines to implement various steps of Sphere methods 1, 2, and test their performance on some randomly generated problems. We use the MATLAB function "*randn*" that generates random numbers from the Standard Normal distribution to generate entries of the coefficient matrix, and vector c. To ensure feasibility of the LP generated, we use the function "$-rand$" that generates random numbers from the Uniform distribution in $(-1, 0)$ for the vector b. To ensure boundedness we include box constraints $l \leq x \leq u$, where l and u are n-vectors with negative and positive random entries respectively.

We run our test problems on a laptop computer with Intel(R) Pentium(R) M processor 2.00GHz and 2.00 GB of RAM.

	Sphere Method 1					Sphere Method 2				
r	n_{fn}	n_{pd}	δ	cx^{r+1}	t_r	n_{fn}	n_{pd}	δ	cx^{r+1}	t_r
1	72	49	2.1251	-0.4107e+3	7.2	13	25	2.1265	-0.6612e+3	2.4
2	61	50	1.9469	-0.7898e+3	6.7	19	39	1.7077	-0.9782e+3	2.9
3	34	50	1.3899	-1.0505e+3	5.3	3	41	1.0402	-1.1650e+3	2.6
4	28	39	0.8184	-1.2056e+3	4.1	1	40	0.5175	-1.2576e+3	2.2
5	17	42	0.3751	-1.2783e+3	4.7	1	42	0.2274	-1.2985e+3	2.3
6	10	34	0.1500	-1.3077e+3	3.5	1	39	0.0925	-1.3152e+3	2.1
7	10	34	0.0594	-1.3194e+3	3.5	1	31	0.0386	-1.3225e+3	1.6
8	0	26	0.0231	-1.3249e+3	2.5	0	14	0.0139	-1.3255e+3	0.8
9	0	4	0.0028	-1.3256e+3	0.9	0	9	0.0045	-1.3266e+3	0.5
10	0	4	0.0009	-1.3259e+3	0.9	0	3	0.0009	-1.3268e+3	0.2
11	0	1	0.0005	-1.3261e+3	0.2	0	2	0.0003	-1.3270e+3	0.1
12	0	1	0.0004	-1.3262e+3	0.2	0	1	0.0000	-1.3271e+3	0.0

Table 1: Comparison of Sphere methods 1 and 2 on a random problem with $m = 150$ and $n = 50$

Table 1 compares the results of implementing Sphere methods 1 and 2 on a randomly generated problem with 50 variables and 150 constraints. In this table n_{fn} and n_{pd} show the number of calls to LSFN and LSCPD respectively, δ is the final value of the radius of the largest sphere inscribed in the feasible region, cx^{r+1} is the objective value at the end of iteration r, and t_r is the cpu time (in seconds) of iteration r.

The results of this table clearly show the superiority of Sphere method 2. The most expensive step in sphere methods is the centering procedure. Table 1 shows that the number of calls to LSFN and LSCPD is substantially reduced in Sphere method 2. Moreover, the new centering strategies discussed in Sphere method 2 make LSFN and LSCPD more efficient which is the reason that the cpu time of each iteration of Sphere Method 2 is significantly lower than that of Sphere Method 1.

Furthermore, the new centering strategies along with the new descent step D5.2 result in higher reduction in objective value in each iteration, especially in early iterations. This can be seen by comparing the objective values at the end of the first iteration. Both methods start

r	cx^r	$cx_{d^1}^{r+1}$	$cx_{d^2}^{r+1}$	$cx_{d^3}^{r+1}$	$cx_{d^4}^{r+1}$	$cx_{d^{5.1}}^{r+1}$	d	cx^{r+1}
1	0	-33.0105	-3.4812	-37.4422	-33.8997	-36.5396	$D3$	-38.6064
2	-38.6064	-65.3938	-62.4668	-66.4183	-65.5638	-67.0769	$D5.1$	-67.5056
3	-67.5056	-74.6753	-76.8942	-75.4481	-74.7344	-75.8154	$D2$	-77.7214
4	-77.7214	-87.1767	-79.4884	-88.4101	-87.4187	-87.7793	$D3$	-88.6782
5	-88.6782	-93.8010	-90.2221	-94.0660	-93.8170	-94.6348	$D5.1$	-94.6580
6	-94.6580	-97.2274	-96.0250	-97.5224	-97.2404	-97.5320	$D5.1$	-97.5517
7	-97.5517	-98.9134	-97.8257	-99.0256	-98.9192	-99.2543	$D5.1$	-99.3225
8	-99.3225	-99.6955	-99.4039	-99.7263	-99.6970	-99.8143	$D5.1$	-99.8180
9	-99.8180	-100.0458	-99.8597	-100.1033	-100.0468	-100.0808	$D3$	-100.1229
10	-100.1229	-100.2347	-100.1355	-100.2496	-100.2368	-100.2770	$D5.1$	-100.3031
11	-100.3031	-100.3269	-100.3046	-100.3324	-100.3271	-100.3346	$D5.1$	-100.3376
12	-100.3376	-100.3376	-100.3376	-100.3376	-100.3376	-100.3378	$D5.1$	-100.3378

Table 2: Descent steps in Sphere method 2 based on D1 to D5.2 on a random problem with $m = 500$ and $n = 50$

from zero as an initial objective value. At the end of iteration 1, Sphere method 2 reduces the objective value 60% more than that of Sphere method 1. Although such superiority is not observed subsequently, but the objective value of Sphere method 2 is better throughout the algorithm.

Notice that at most iterations the value of δ in Sphere method 2 is smaller than that of Sphere method 1. This is due to the fact that at each iteration the approximate ball center obtained in Sphere method 2 is closer to the optimum solution of the original LP and its updated feasible region gets smaller as the algorithm progresses.

Table 2 shows the results of descent steps D1 through D5.2 of Sphere method 2 on a randomly generated problem with 50 variables and 500 constraints. This instance problem is solved in 12 iteration. In this table cx^r is the objective value in the beginning of iterations r, $cx_{d^i}^{r+1}$, for $i = 1, , 2, 3, 4, 5.1$, illustrate the result of the objective value at the point obtained from D1 to D5.1, d is the descent step that produced the best point among D1 to D5.1, and the last column

Variables n	Constraints m	Sphere Method 1 cpu time (sec.)	Sphere Method 2 cpu time (sec.)	Simplex Method cpu time (sec.)
50	500	43	27	74
	1000	40	29	122
	1500	42	32	360
100	700	136	108	241
	1200	175	133	524
	1700	342	256	892
200	900	1297	973	1587
	1200	1091	892	3520
	2000	1180	923	4921
300	1800	1721	1237	7590
	2500	1380	1020	9884
	3000	1232	951	9999

Table 3: Comparison of CPU time for randomly generated problems with $m >> n$ for Sphere methods 1 and 2 and Simplex method.

shows the result of the descent step after implementing D5.2 at the end beginning with the best point obtained from D1 to D5.1. Since D5.2 can be carried out very cheaply we implement this step at the end of the descent step. Notice that D5.2 either improves the objective value or doesn't change it. Therefore the result of D5.2 is indeed the final value of the objective function at the end of iteration r, that is cx^{r+1}. Observe that in most of the iterations descent step D5.1 gave the best point among D1 to D5.1, and D5.2 slightly improves the objective value.

In Table 2, observe that at the end of the first iteration, Sphere method 2 improves the current point by about 40% (from 0 to -38.6064), and by the end of the fifth iteration a 95% improvement is achieved. We tried several examples and the same behavior was more or less observed in all instances where there exist too many constraints. This observation suggests that Sphere method 2 is particularly efficient in problems where $m >> n$.

In Tables 1 and 2 we showed that Sphere method 2 has superiority over Sphere method 1 in terms of number of iterations in centering procedure as well as the objective value at each iteration. We now show that Sphere method 2 also outperforms Sphere method 1 and Simplex method in terms of overall cpu time.

Table 3 compares the cpu time (in second) of Sphere methods 1 and 2 and the Simplex method. We tested on random problems with moderate number of variables (up to 300) and large number of constraints (up to 3000). The first two columns of the table illustrate the number of variables and the number of constraints respectively, and the last three columns report the cpu time of Sphere method 1 and 2 and simplex method respectively. The results of the test problems reported in this table suggest that the improvement of Sphere method 2 over Sphere method 1 varies between 20% to 40%. This observation is very encouraging and it shows that the combination our new centering strategy and descent steps discussed in this paper works well in practice.

5 Conclusion

We presented some preliminary computational results on implementing Sphere Methods 1, 2 by solving each step in these methods using MATLAB 7.0 routines separately; and compared this performance with that of MATLABs finished LP code "linprog" based on the simplex method. The results show that even this implementation of the sphere methods performs much better than "linprog".

To compare the sphere methods with existing IPMs will require developing a low-level programming language code for them using advanced techniques of numerical linear algebra, and updating the basis inverse in LSCPD steps as the matrix grows by a row and column as described above; which we have not done in these preliminary experiments. But these preliminary results, and the fact that the work in each iteration of Sphere method 2 is much simpler than an iteration of other IPMs indicates that Sphere method 2 will have advantage over them for solving large scale models, in particular when the models may have redundant constraints, or a coefficient matrix that is not very sparse.

6 References

1. G. B. Dantzig and M. N. Thapa, 1997, *Linear Programming, Vol. 1. Introduction , Vol. 2. Theory and Extensions*, Springer-Verlag New York.

2. M. Kojima, S. Mizuno, and A. Yoshise, "A primal-dual interior point algorithm for linear programming", *Progress in Mathematical Programming: Interior Point and Related Methods*, N. Megiddo, ed., Springer-Verlag, New York, ch. 2 (29-47) 1989.

3. N. Megiddo, "Pathways to the optimal set in linear programming", *Progress in Mathematical Programming: Interior Point and Related Methods*, N. Meggiddo, ed., Springer-Verlag, New York, ch. 8 (131-158) 1989.

4. S. Mehrotra, "On the implementation of a primal-dual interior point method", *SIAM Journal on Optimization*, 2 (575-601) 1992.

5. R. D. C. Monteiro and I. Adler, "Interior path-following primal-dual algorithms, Part I: Linear programming", *Mathematical Programming* 44 (27-41) 1989.

6. K. G. Murty, and M. R. Oskoorouchi, "Note on implementing the new sphere method for LP using matrix inversions sparingly", *Optimization Letters*, 3, 1, November 2008, 137-160.

7. K. G. Murty, "A new practically efficient interior point method for LP", *Algorithmic Operations Research*, 1 (3-19) 2006; paper can be seen at the
website: http://journals.hil.unb.ca/index.php/AOR/index.

8. K. G. Murty, "Linear equations, Inequalities, Linear Programs (LP), and a New Efficient Algorithm" Pages 1-36 in *Tutorials in OR*, INFORMS, 2006.

9. K. G. Murty, and S. N. Kabadi, "Additional Descent Steps in the Sphere Method", Dept. IOE, University of Michigan, Ann Arbor (2008)

10. R. Saigal. *Linear Programming A Modern Integrated Analysis.* Kluwer Academic Publishers, Boston, MA, 1995.

11. G. Sonnevend, J. Stoer, and G. Zhao, "On the complexity of following the central path of linear programming by linear extrapolation", *Mathematics of Operations Research* 62 (19-31) 1989.

12. S. J. Wright. *Primal-Dual Interior-Point Methods.* SIAM, Philadelphia, PA, 1997.

13. Y. Ye. *Interior Point Algorithms, Theory and Analysis*, Wiley-Interscience, New York, 1997.