Algorithmic Operations Research Vol.5 (2010) 21–33

Sphere Methods for LP

Katta G. Murty

Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI 48109-2117, USA;
SE Department, King Fahd University of Petroleum and Minerals, Dhahran-31261, Saudi Arabia

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

Abstract

The sphere method for solving linear programs operates with only a subset of constraints in the model in each iteration, and thus has the advantage of handling instances which may not be very sparse. In this paper we discuss enhancements, and improved versions Sphere Methods 2, 2.1 which improve performance by 20 to 80%. Additional steps that can improve performance even more are also presented.

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, 2.1 for large scale LPs.

1. Introduction

Among mathematical models, Linear Programming (LP) is perhaps the most commonly used in decision making applications. The simplex method (Dantzig and Thappa [1997]) 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, 4, 5, 7, 16 to 19]) 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 (Murty and Oskoorouchi [2008]) 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 methods need an initial interior feasible

Email: Katta G. Murty [murty@umich.edu], Mohammad R. Oskoorouchi [moskooro@csusm.edu].

solution. Each iteration of these methods begin 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 this 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.

For any matrix D, we denote its ith row, jth column by $D_{i.}, D_{.j}$ respectively. We use the symbol e to denote a column vector with all entries "1", its dimension appropriate to the context. For any given set Γ of points in R^n, we denote its convex hull by $< \Gamma >$.

The sphere methods consider 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 [11]. 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 [11] 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: $A_{i.}x - b_i$ = 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.

A ball center of K may not be unique; and in case it is not unique, [15] discusses a conceptual definition of identifying a specific one among them to be called "**the ball center of K**". This theoretical definition guarantees that the ball center is well defined and unique for every polytope; 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 $H \cap K^0 \neq \emptyset$. But for computational efficiency in practice, in sphere methods we will use any ball center as defined above, computed approximately.

Reference [15] 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 [15] selects a direction y which is a **profitable direction** for K 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 $\alpha \geq 0$. A direction y has been shown in [10, 11, 15] to be a profitable direction for K 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 $\bar{x} \in K^0$ 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.

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

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, or the linear time algorithm of Megiddo [1983]).

Two procedures for generating profitable directions are discussed in [15], 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.

The aim of this paper is to discuss some enhancements to the sphere method of [15] called Sphere Method 1 here, leading to newer versions, and provide preliminary computational experience with them that shows up to 40% improved performance, and outline some future theoretical and computational work on these methods.

As discussed in [15], if K is unbounded, the ball center of K may not be well defined; but the implementation of the algorithm based on the approximate computation of ball centers can be carried out as usual. If the objective value in (1) is unbounded below, in one of the steps the step length may turn out to be $+\infty$. So, for simplicity we assume that K is bounded in this paper.

2. 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 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.

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. 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., denoting the output of iteration s by x^s, 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)). See [15] for a detailed description of this method.

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.

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 $z = cx$.

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 [15]).

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 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 [15]).

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

$$\begin{aligned} &A_{i.}y = 1 \quad \text{for all} \quad i \in T^{r+1}(\hat{x}) \\ &-cy = \beta \end{aligned} \tag{3}$$

where β is a positive parameter. Earlier (and in sphere method 1, Section 2.1 of [15]) 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 (3) as a function of this parameter β. Let this solution be denoted by $p + \beta q$.

As in Section 2.1 of [15], if B is a basis associated with the basic vector y_B obtained for (3), let y_D denote the vector of remaining nonbasic variables in (3) 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

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

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 1.

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

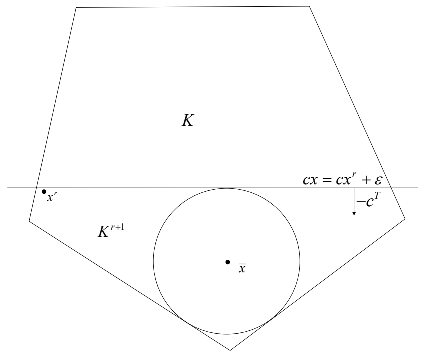

Fig. 1. K is the original set of feasible solutions of the LP being solved. The current set of feasible solutions in an iteration when x^r is the initial interior feasible solution, is K^{r+1}. The ball shown is the largest ball inside K^{r+1}, and its center $\bar{x}$ is a ball center obtained in the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

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 $\delta(\tilde{x}^{r1}) \leq \epsilon$. 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.

Let x^s denote the best point obtained at the end of this cycle of D5.2 steps. With this point go to Descent step 5.3 (D5.3).

D5.3, Descent Step 5.3: We have the current point x^s with $\delta(x^s) \leq \epsilon$.

For each $i \in T(x^s)$, define x^{si} = orthogonal projection of x^s on facetal hyperplane $\{x : A_{i.}x = b_i\}$ = $x^s + (A_{i.})^T(b_i - A_{i.}x^s)$. Define

$\bar{x} = [\sum_{i\in T(x^s)} x^{si}]/|T(x^s)|$. Typically, a move from x^s in the direction $x^s - \bar{x}$ goes through the central portion of K^{r+1}, so a step in this direction at this stage can be expected to lead to good improvement in objective value. We have 2 cases to consider.

Case 1: If $c(x^s - \bar{x}) < 0$ carry out a descent step at x^s in the descent direction $(x^s - \bar{x})$, and make the output of this descent step the new current point (new x^s) and repeat this step with it, as long as the improvement in objective value is greater than the selected tolerance.

Case 2: If $c(x^s - \bar{x}) \geq 0$, let y be the orthogonal projection of $(x^s - \bar{x})$ on the hyperplane $\{x : cx = 0\}$, $y = (I - c^T c)(x^s - \bar{x})$.

Solve the 2-variable LP: max δ subject to $\delta - A_{i.}y \leq A_{i.}x^s - b_i$ for i = 1 to m, and $\delta, \alpha \geq 0$. Let $\bar{\delta}, \bar{\alpha}$ be the optimum solution of this 2-variable LP. The point $x^s + \bar{\alpha}y$ has objective value = cx^s because $cy = 0$, from this point take all descent steps D1 up to D5.2. Call the final output point of these descent steps as the new current point (new x^s), and with it repeat this D5.3 until the improvement in objective value becomes less than the selected tolerance.

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 [15].

Instead of giving β the specific value 1 as in earlier methods, leaving it as a positive parameter in (4), improves the performance of the centering step in sphere method 2.

4. Preliminary Computational Results

In this section we present the results of our computational tests on sphere methods 1 and 2. Broadly these results indicate that sphere method 2 is up to 80% faster 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 dense problems. We use the MATLAB function "$randn$" that generates random numbers from the Standard Normal distribution to generate entries of the dense 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 reset the random number generator to a different state each time we use "rand('state',sum(100*clock))". 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.

4.1. *Computational results*

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 that sphere method 2 has improved performance. 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 steps D5.2 and D5.3 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 from zero as an initial objective value. At the end of iteration 1, sphere method 1 reduces the objective value to -410.7 while sphere method 2 reduces the objective value to -748.6. That is over 80% improvement in reducing the objective value. Although such improved behavior is not observed subsequently, but the objective value of sphere method 2 is better throughout the algorithm, which results in faster convergence to a more accurate solution.

Notice that in 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

Table 1

	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	-0.7486e+3	2.3
2	61	50	1.9469	-0.7898e+3	6.7	18	37	1.6155	-1.0215e+3	2.5
3	34	50	1.3899	-1.0505e+3	5.3	4	39	0.9856	-1.1803e+3	2.1
4	28	39	0.8184	-1.2056e+3	4.1	1	38	0.4955	-1.2963e+3	2.1
5	17	42	0.3751	-1.2783e+3	4.7	1	36	0.2096	-1.3220e+3	2.0
6	10	34	0.1500	-1.3077e+3	3.5	1	32	0.0648	-1.3252e+3	1.9
7	10	34	0.0594	-1.3194e+3	3.5	1	27	0.0198	-1.3261e+3	1.2
8	0	26	0.0231	-1.3249e+3	2.5	0	9	0.0095	-1.3269e+3	0.7
9	0	4	0.0028	-1.3256e+3	0.9	0	5	0.0008	-1.3270e+3	0.2
10	0	4	0.0009	-1.3259e+3	0.9	0	1	0.0001	-1.3272e+3	0.1
11	0	1	0.0005	-1.3261e+3	0.2					
12	0	1	0.0004	-1.3262e+3	0.2					

Comparison of sphere methods 1 and 2 on a random dense problem with $m = 150$ and $n = 50$

Table 2

r	cx^r	cx^{r+1}_{d1}	cx^{r+1}_{d2}	cx^{r+1}_{d3}	cx^{r+1}_{d4}	$cx^{r+1}_{d5.1}$	d	cx^{r+1}
1	0	-33.2598	-3.5691	-37.5625	-33.2285	-36.6348	$D3$	-40.2655
2	-40.2655	-65.2287	-63.0900	-67.01446	-65.2355	-67.4223	$D5.1$	-68.0023
3	-68.0023	-75.02559	-76.4776	-75.2263	-74.8651	-75.3991	$D2$	-78.9358
4	-78.9358	-87.1765	-79.4524	-88.0251	-87.4199	-87.9025	$D3$	-89.4825
5	-89.4825	-93.8825	-90.6598	-94.5022	-93.0255	-94.7996	$D5.1$	-95.1169
6	-95.1169	-97.2548	-96.0556	-97.0846	-97.8524	-97.9056	$D5.1$	-98.0005
7	-98.0005	-98.9134	-97.8257	-99.0256	-98.9192	-99.2543	$D5.1$	-99.9903
8	-99.9903	-100.0569	-100.0055	-100.1122	-100.0955	-100.0255	$D3$	-100.2965
10	-100.2965	-100.3055	-100.3144	-100.3336	-100.3289	-100.3356	$D5.1$	-100.3379

Descent steps in sphere method 2 based on D1 to D5.3 on a random dense problem with $m = 500$ and $n = 50$

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 D5.3 of sphere method 2 on a randomly generated dense problem with 50 variables and 500 constraints. This instance problem is solved in 10 iteration. In this table cx^r is the objective value in the beginning of iterations r, $cx^{r+1}_{d^i}$, 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 shows the result of the descent step after implementing D5.2 and then D5.3 beginning with the best point obtained from D1 to D5.1. Notice that we only report the final result after implementing D5.3. Since D5.2 and D5.3 can be carried out very cheaply we implement these steps at the end of the descent step. Notice that the result of D5.3 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 and then D5.3 slightly improve the objective value.

In Table 2, observe that at the end of the first iteration, sphere method 2 improves the current point by over 40% (from 0 to -40.2655), and by the end of the fifth iteration a 95% improvement is achieved. We tried several examples where there exist too many constraints, and more or less the same behavior was observed in all instances. This observation suggests that sphere method 2 is particularly efficient in problems where $m \gg n$.

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. We use the Matlab function "linprog" with the simplex option to solve the instance problems by the simplex method.

Table 3 compares the cpu time (in seconds) of sphere methods 1 and 2 and the simplex method. We tested on randomly generated fully dense 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 50%. This observation is very encouraging as it shows that the combination of our new centering strategy and descent steps discussed in this paper works well in practice.

As it was stated earlier, a substantial portion of the cost at each iteration belongs to the centering procedure. Next we discuss techniques for improving the implementations of this centering procedure.

4.2. *Computational results of light centering*

Of the two steps in an iteration of sphere method 2, the centering step is computationally most expensive, compared to which the descent step is very cheap. The centering step is expensive because in order to get a good approximation of a true ball center of K^{r+1} takes several line search steps in the LSFN, LSCPD sequences.

One of the great advantages of sphere methods over other IPMs for LP for practical implementations, is the flexibility that they offer in implementing them. In implementing sphere methods, the expensive centering step need not be carried out to the full extent described in the statement of the algorithm in every iteration.

Actually, a true ball center is not really necessary to continue applying the algorithm. Stopping the work in a centering step short of getting a very good approximation to a ball center of K^{r+1} leads to what we will call a **subiteration based on light centering**. In this subiteration we select a small positive integer ℓ (something like 4 to 10), and carry out only ℓ line search steps in each of the LSFN, LSCPD sequences in that order, and take the point obtained at the end as an approximate ball center to move over to the descent step.

To incorporate these efficiently, we do the following in an iteration: the iteration begins with a sequence of subiterations, each consisting of a light centering step as defined above, followed by a descent step as discussed in earlier sections. The point obtained at the end of a subiteration is the initial point for the next subiteration. This process is continued as long as the improvement in objective value in each subiteration is greater than a selected tolerance. When the objective value improvement in a subiteration falls below the tolerance, carry out a full iteration based on the full centering step as discussed in Section 3 for sphere method 2, and then the descent step. The best point obtained at the end of all these descent steps is the output of this iteration, with

Table 3

Variables n	Constraints m	Sphere Method 1 cpu time (sec.)	Sphere Method 2 cpu time (sec.)	Simplex Method cpu time (sec.)
50	500	43	22	74
	1000	40	24	122
	1500	42	28	360
100	700	136	99	241
	1200	175	121	524
	1700	342	236	892
200	900	1297	933	1587
	1200	1091	815	3520
	2000	1180	901	4921
300	1800	1721	1125	7590
	2500	1380	932	9884
	3000	1232	891	9999

Comparison of CPU time for randomly generated fully dense problems with $m \gg n$ for sphere methods 1 and 2 and simplex method.

which the algorithm moves to the next iteration.

We now show some improvements on the computational results with the light centering technique. Let us call this version, sphere method 2.1. Table 4 compares the objective values in each iteration of sphere methods 2 and 2.1 for a randomly selected problem with 50 variables and 600 constraints. For sphere method 2 we report the objective values at three stages in each iteration: the center obtained by LSFN, the center obtained by LSCPD, and the end of descent steps. For sphere method 2.1 we report the objective values at the points obtained by the light centering procedures of LSFN and LSCPD, the full centering of LSFN and LSCPD, and the end descent steps in each iteration. We also report the cpu time of each iteration.

Each iteration of sphere method 2.1 involves several light centering cycles and one full centering cycle. Table 4 shows that the reduction in the objective function value in an iteration of sphere method 2.1 is substantially more than that of sphere method 2, especially in earlier iterations which results in fewer overall iterations. Although the cpu time per iteration in sphere method 2.1 is slightly higher than that of sphere method 2, but the significant reduction in the number of iteration more than makes up for this in the overall cpu time.

In Table 5 we repeat the results of Table 3 with an additional column for the total cpu time of sphere method 2.1. This table shows that sphere method 2.1 with the light centering procedure improves the computational results of sphere method 2.

As mentioned earlier the computational results reported in this paper are based on a MATLAB code that uses built-in functions. This was mainly done to obtain quick computational results. We are aware of the fact that MATLAB is not recommended for computational algorithms with many loops such as sphere methods. Therefore the computational results reported here are only preliminary. Converting some of the subroutines written for this code to a lower level language using MEX files, or rewriting the whole code, would significantly reduce its cpu time.

The computational results reported in this section suggest that sphere methods are efficient methods for dense problems where $m \gg n$. A professional implementation of these methods is required to test the real power of these techniques. We believe such implementation would make sphere methods an alternative and competitive techniques to primal-dual interior point methods.

5. How to Improve the Performance of Sphere Methods Even Further?

Since the centering step is computationally the most expensive step in each iteration, the key to improving the performance of sphere methods lies in reducing the need for the centering steps in the method.

Let $\{\hat{x}^1, ..., \hat{x}^s\}$ denote the set of all points obtained at the end of the various descent steps in D5.1 in an iteration, and K_1 its convex hull. $s \leq m$, and typically $s \leq n + 1$. Also, $\hat{x}^1, ..., \hat{x}^s$ are spread out in different directions all around K, each one in the interior of K but close to the boundary of K. So, intuitively, it seems that a ball center for K_1 may be close to a ball center

Table 4

	Sphere Method 2				Sphere Method 2.1					
r	cx_{fn}	cx_{pd}	cx^{r+1}	t_r	cx_{fn}^{light}	cx_{pd}^{light}	cx_{fn}	cx_{pd}	cx^{r+1}	t_r
1	-0.2431	-1.1644	-3.0553	2.3	-0.3758	-0.9625	-1.2091	-1.9222	-3.4269	2.7
2	-3.1453	-3.2994	-4.9102	2.3	-3.5218	-4.2662	-4.2956	-4.2592	-5.1072	2.8
3	-4.9830	-5.0040	-5.8053	2.4	-5.0418	-5.1567	-5.1614	-5.3005	-5.9035	2.2
4	-5.8224	-5.8072	-5.9554	2.1	-5.8204	-5.9869	-5.9909	-5.9915	-6.2436	2.0
5	-5.9605	-5.9667	-6.2281	2.2	-6.1728	-6.2533	-6.2533	-6.2581	-6.3990	0.2
6	-6.2376	-6.2651	-6.3688	2.2						
7	-6.3691	-6.3689	-6.3790	2.1						
8	-6.3805	-6.3810	-6.3900	1.9						
9	-6.3909	-6.3922	-6.3966	0.5						
10	-6.3971	-6.3975	-6.3989	0.1						

Comparison of objective values of sphere methods 2 and 2.1 with $m = 600$ and $n = 50$

Table 5

n	m	Sphere Method 1 cpu time (sec.)	Sphere Method 2 cpu time (sec.)	Sphere Method 2.1 cpu time (sec.)	Simplex Method cpu time (sec.)
	500	43	22	9	74
50	1000	40	24	10	122
	1500	42	28	11	360
	700	136	199	23	241
100	1200	175	121	32	524
	1700	342	236	48	892
	900	1297	933	215	1587
200	1200	1091	815	279	3520
	2000	1180	901	356	4921
	1800	1721	1125	1001	7590
300	2500	1380	932	1109	9884
	3000	1232	891	1223	9999

Comparison of CPU time for randomly generated problems with $m \gg n$ for sphere methods 1, 2 and 2.1 and simplex method.

for K on the objective plane through it. So, if we can obtain an approximation to the ball center of K_1 directly and efficiently, we may be able to use it as the center for the next iteration, and thus saving the need for a centering step. We will now present some results from Murty [2009-1] in this respect.

There are two ways of representing a convex polytope. One way represents it as the set of feasible solutions of a given system of linear constraints, this is the representation we used in Section 1 to define the set of feasible solutions of the LP (1). Another way is to represent the polytope as the convex hull of a given set of points, the polytope K_1 whose ball center we want to obtain is represented this way as the convex hull of $\Gamma = \{\hat{x}^1, ..., \hat{x}^s\}$.

For carrying out the various steps in sphere method 1 for the LP (1), we needed the solutions of the following problems.

Problem 1: Given any interior point x in the polytope K, what is the radius $\delta(x)$ of the largest ball with x as center, that is contained inside K?

Problem 2: We want to find a largest ball inside the polytope K. Is it unique, and if so what is its center and radius? If not, what is the set of all points, each of which is the center of a largest ball inside K? Also, given an interior point of K, how can we check whether it is the center of a largest ball inside K?

Problem 3: Given an interior point $\bar{x}$ of K, a $y \in R^n,\ y \neq 0$ is said to be a **profitable direction** at $\bar{x}$, if $\delta(\bar{x} + \alpha y)$ increases strictly as α increases from 0. How can we check efficiently whether a given $y \neq 0$ is a profitable direction at $\bar{x}$? How can we check whether there exists a profitable direction at

$\bar{x}$, and if so how can we compute one such direction efficiently?

Consider these 3 problems for the polytope K defined by the system of linear constraints in (1). As defined in Section 1, $\delta(x)$ of Problem 1 is minimum $\{A_{i.}x - b_i : i$ = 1 to $m\}$, can be computed very efficiently. Let δ be the radius of a largest ball inscribed in K, and x its center; then (x, δ) is an optimum solution of the LP: maximize δ subject to $\delta \leq A_{i.}x - b_i$, i = 1 to m; and vice versa; and finally the ball center of K is unique iff this LP has a unique optimum solution; hence Problem 2 can be solved efficiently. As discussed in Section 1, y is a profitable direction at $\bar{x}$ iff $A_{i.}y > 0$ for all $i \in T(\bar{x})$ where the definition of $T(\bar{x})$ is given in Section 1. Hence all 3 problems can be solved efficiently for the polytope K defined by the system of linear constraints (1).

However, there are no efficient methods known to solve any of Problems 1, 2, 3 listed above on a polytope P represented as the convex hull of a given set of points. For instance, Theorem 1 below shows that Problem 1 for the polytope P is NP-hard.

Theorem 1: $P = < \{\tilde{x}^1, ..., \tilde{x}^L\} >$, is a polytope of dimension n in R^n given in the convex hull representation; and x^0 is a given interior point of P. The problem of computing $\tilde{\delta}(x^0)$ = the radius of the largest ball inscribed in P with x^0 as center, is NP-hard.

Proof: Transfer the origin to x^0. Then P becomes $< \tilde{\tilde{x}}^1, ..., \tilde{\tilde{x}}^L\} >$, where $\tilde{\tilde{x}}^t = \tilde{x}^t - x^0$. In this representation, since $\tilde{x}^0 = 0$ is an interior point of P, P can be represented by a system of linear inequalities of the form $\tilde{D}x \leq e$; where e is a column vector of all 1s, and $\tilde{D}$ is a matrix of order $m_1 \times n$; m_1 being the number of facets of P.

Then for each for each i = 1 to m_1, $\tilde{D}_{i.}$ is the i-th row vector of $\tilde{D}$, and $\{x : \tilde{D}_{i.}x = 1\}$ is a facetal hyperplane of P. So the Euclidean distance between $\tilde{x}^0 = 0$, and the nearest point to $\tilde{x}^0$ on this facetal hyperplane is $1/||\tilde{D}_{i.}||$. So, $\tilde{\delta}(x^0)$ = minimum$\{1/||\tilde{D}_{i.}||$: i = 1 to $m_1\}$.

The vectors $\tilde{D}_{i.}$, i = 1 to m_1 are all the extreme points of the system:

$$a\tilde{\tilde{x}}^t \leq 1 \quad \text{for all } t = 1 \text{ to } L \tag{4}$$

in variables $a = (a_1, \ldots, a_n)$. Hence $\tilde{\delta}(x^0)$ is the inverse of the optimum objective value in the problem of maximizing $||a||$ subject to the system of inequalities (4). This is the problem of maximizing the Euclidean norm, a convex function, on a convex polytope specified by a system of linear inequalities, a well known NP-hard problem. □

However we noticed that typically the set $\Gamma = \{\hat{x}^1, ..., \hat{x}^s\}$ is either linearly independent, or forms a simplex when $s = n + 1$. Murty [2009-1] shows that in these cases, K_1 has a unique ball center which can be computed directly. We discuss those results next.

Ball Center of a Simplex Represented by Constraints

Let S be an n-dimensional simplex in R^n, i.e., its representation using linear constraints is of the form

$$S = \{x : D_{i.} \geq d_i \quad \text{for } i = 1 \text{ to } n + 1\}$$

where $d = (d_i) \in R^{n+1}$, and the coefficient matrix D of order $(n + 1) \times n$ with rows $D_{i.}$ i = 1 to $n + 1$ satisfies the properties that all the $(n + 1)$ submatrices of it of order $n \times n$ are nonsingular, and that each row vector of D is a linear combination of the other rows with strictly negative coefficients. Without any loss of generality we will assume that all the rows of D have been normalized so that $||D_{i.}||$ = 1 for all i.

We will now show that S has a unique ball center which is the unique solution of the system of linear equations:

$$Dx - \delta e = d \tag{5}$$

where x in the solution will be the ball center of S, and δ is the radius the largest ball inside S with center at that x.

It can be verified that the coefficient matrix of this system is nonsingular, hence this system has a unique solution, $(\bar{x}, \bar{\delta})$. From (5) we know that $\bar{\delta} = D_{i.}\bar{x} - d_i$ for all i = 1 to $n + 1$, so that the ball $B(\bar{x}, \bar{\delta})$ with $\bar{x}$ as the center and $\bar{\delta}$ as radius is inside S, and touches all the facets of S, so it is the largest ball with $\bar{x}$ as center inside S.

Also, the system $Dy \geq 0$ has 0 as its unique solution, because $D_{(n+1).} = \alpha_1 D_{1.} + ... + \alpha_n D_{n.}$ with all $\alpha_1, ..., \alpha_n < 0$. So, there is no profitable direction for the polytope S at $\bar{x}$, this implies that $\bar{x}$ is the ball center of S and that it is unique.

So, when the simplex S is represented using linear constraints, its ball center can be computed efficiently by solving the system of linear constraints (5).

Ball Center of a Simplex Represented as the Convex Hull of its Extreme Points

Now suppose that S is represented as the convex hull of its set of extreme points $\Gamma = \{x^1, ..., x^{n+1}\}$. So, an $x \in S$ is represented as:

$$x = \beta_1(x^1 - x^{n+1}) + ... + \beta_n(x^n - x^{n+1}) + x^{n+1} \quad (6)$$

where $\beta_1, ..., \beta_n \geq 0$, and $\beta_1 + ... + \beta_n \leq 1$.

Let B denote the $n \times n$ matrix with its j-th column $B_{.j} = x^j - x^{n+1}$, for j = 1 to n; and $\beta = (\beta_1, ..., \beta_n)^T$. Since S is a simplex we know that B is nonsingular. So, from (6), we have

$$B^{-1}x = \beta + B^{-1}x^{n+1}.$$

Using this, from the bounds on β we have

$$\begin{aligned} B^{-1}x &\geq B^{-1}x^{n+1} \\ -eB^{-1}x &\geq -1 - eB^{-1}x^{n+1} \end{aligned} \quad (7)$$

So, (7) is the representation of S = convex hull of Γ here through linear constraints. To derive the ball center of S using this representation (7), we need to normalize each constraint in (7) so that the Euclidean norm of its coefficient vector is 1. For this we need $\gamma_i = ||(B^{-1})_{i.}||$ for i = 1 to n, and $\gamma_{n+1} = ||eB^{-1}||$. Then from the results discussed above we know that the ball center x of S, and the radius δ of the largest ball inside S, are the solution of the system

$$\begin{pmatrix} B^{-1} & -\gamma \\ -eB^{-1} & -\gamma_{n+1} \end{pmatrix} \begin{pmatrix} x \\ \delta \end{pmatrix} = \begin{pmatrix} B^{-1}x^{n+1} \\ -1 - eB^{-1}x^{n+1} \end{pmatrix}$$

where $\gamma = (\gamma_1, ..., \gamma_n)^T$. By adding the sum of the first n equations in this system to the last, we find that in the solution of this system

$$\delta = 1/(\gamma_1 + ... + \gamma_{n+1})$$

and from the first n equations we see that the ball center of the simplex S here is

$$x = x^{n+1} + (1/(\gamma_1 + ... + \gamma_{n+1}))B\gamma$$

Ball Center of the Convex Hull of a Linearly Independent Set of Vectors in R^n

Let $Q = \{x^1, ..., x^r\}$ be a linearly independent set of column vectors in R^n, and S_2 = convex hull of Q, where $r \leq n$. Then S_2 is an $(r-1)$-dimensional simplex in its affine hull. In this section we discuss how to compute the ball center of S_2 directly.

In this case the $n \times (r-1)$ matrix

$$B_2 = (x^1 - x^r \vdots \cdots \vdots x^{r-1} - x^r)$$

is of full column rank. Find a row partition of it into $(B_{21}, B_{22})^T$ such that B_{21}, B_{22} are of orders $(r-1) \times (r-1)$ and $(n-r+1) \times (n-r+1)$ respectively, and B_{21} is nonsingular.

Let $(\underline{x}_1^j, \underline{x}_2^j)^T$, $(\underline{x}_1, \underline{x}_2)^T$ be the corresponding row partitions of the column vectors x^j for each j = 1 to r, and each $x \in S_2$ respectively.

Then for each $x = (\underline{x}_1, \underline{x}_2)^T \in S_2$, it can be verified that

$$\underline{x}_2 = B_{22}[B_{21}^{-1}(\underline{x}_1 - \underline{x}_1^r)] + \underline{x}_2^r \quad (8)$$

(8) is the system of linear equations that defines the affine hull of S_2.

Now, the convex hull of $\{\underline{x}_1^1, \ldots, \underline{x}_1^r\} \subset R^{r-1}$ is a full dimensional simplex in R^{r-1} and its ball center $\underline{x}_1$ can be found by applying the formula derived earlier to it. Then in the original space R^n, the ball center of S_2 is $(\underline{x}_1, \underline{x}_2)^T$ where $\underline{x}_2$ is obtained from $\underline{x}_1$ using (8).

Application in sphere Methods for Large Scale LPs

As mentioned at the beginning of this section, the set $\Gamma = \{\hat{x}^1, ..., \hat{x}^s\}$ of output points obtained in the various descent steps in D5.1 in an iteration, is typically either linearly independent, or is a simplex, and hence the above results can be used to obtain the ball center of K_1 = convex hull of Γ directly; or we can even carry this out approximately to get an approximate ball center of K_1; from which we can carry out all the descent steps continuing the iteration. The computational performance of this technique needs to be checked.

6. An Open Research Problem in Geometry

We are investigating the worst case computational complexity of the sphere methods. In this investigation, the central problem that appears is the following geometric one.

Let $K = \{x : Ax \geq b\}$ where A is a matrix of order $m \times n$ satisfying: $||A_{i.}|| = 1$ for all i = 1 to m, be a given polytope of full dimension in R^n; and cx a linear objective function with $||c|| = 1$. Let t_{max}, t_{min} be the maximum and minimum values of cx over K with $t_{max} > t_{min}$.

As before, for each $x \in K$, let $\delta(x) =$ minimum$\{A_{i.}x - b_i : i = 1 \text{ to } m\}$, it is the radius of the largest ball that can be inscribed in K with x as center. And the index set $T(x) = \{i : i$ ties for the minimum in the definition of $\delta(x)$ above; it is the index set of all the constraints defining K that are tangent planes to the ball $B(x, \delta(x))$.

For each $t_{max} \geq t \geq t_{min}$ let $H_t = \{x : cx = t\}$. Finding a largest ball inscribed inside K with its center on H_t, is the problem of maximizing $\delta(x)$ over $K \cap H_t$ which is the following:

$$\begin{array}{lll} \text{Maximize} & \delta & \\ \text{subject to} & \delta - A_{i.}x \leq -b_i & \text{for all } i = 1 \text{ to } m \\ & cx = t & \end{array} \tag{9}$$

a parametric RHS (right hand side) LP with t as the parameter. Let $x(t), \delta(t)$ denote an optimum solution to this problem as a function of the parameter t. Then $x(t)$ is the center of a largest ball in K with its center on H_t, and $\delta(t) = \delta(x_t)$ is its radius.

For each t, $\delta(t)$ is unique, but in general $x(t)$ may not be unique if the parametric RHS LP given above is dual degenerate. But to keep our discussion simple (and without loss of generality), we make the assumption that $x(t)$ is also unique for all t; this is a nondegeneracy assumption.

From the results on parametric RHS LPs (e.g., Murty [1980, 1983]), we know that $\delta(t)$ is a piecewise linear concave function, that the range $[t_{min}, t_{max}]$ is partitioned into a finite number of intervals such that in each interval the slope of $\delta(t)$ is constant (so, it is linear in this interval), and as t increases moving from one interval to the next the slope strictly degreases. Also, $T(x(t))$ remains the same for all t in any interval; and this index set of touching constraints changes as t moves from one interval to the next. So the number of these intervals is the number of changes in the index set of touching constraints to the largest inscribed ball in K with center on H_t, as t decreases from t_{max} to t_{min}.

The results in Murty [1980] show that for parametric RHS LPs in general, the number of slope changes in the optimum objective value function can grow exponentially with the size of the problem. But for (9) with its special structure, there are intuitive reasons to believe that the number of slope changes in $\delta(t)$ (equivalently, changes in the touching constraint index set $T(x(t))$ as t decreases from t_{max} to t_{min}) is bounded above by a polynomial in m. In [12] I included a proof to show that this number is bounded above by m, but Mirzaian [6] found an error in my proof, and this problem now remains open. We continue studying it.

7. Conclusion

We presented some preliminary computational results on implementing sphere Methods 1, 2, 2.1 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 methods 2, 2.1, is much simpler than an iteration of other IPMs indicates that sphere methods 2, 2.1 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.

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, 1989, "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).

[3] N. Megiddo, 1983, "Linear-Time Algorithms for Linear Programming in R^3 and related problems", SIAM J. of Computing, 4(4)759-776.

[4] N. Megiddo, 1989, "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).

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

[6] A. Mirzaian, 2007, Private communication.

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

[8] K. G. Murty, 1980, "Computational Complexity of Parametric Linear Programming", Mathematical Programming, 19 (213-219).

[9] K. G. Murty, 1983, *Linear Programming*, Wiley, NY.

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

[11] K. G. Murty, 2006-2, "Linear equations, Inequalities, Linear Programs (LP), and a New Efficient Algorithm" Pages 1-36 in *Tutorials in OR*, INFORMS.

[12] K. G. Murty, 2009-1, "Ball Centers of Special Polytopes", IOE Dept., University of Michigan, Ann Arbor, MI-48109-2217.

[13] K. G. Murty, 2009-2, "New Sphere Methods for LP", *Tutorials in OR*, INFORMS.

Received 20-1-2009; revised 6-9-2009; accepted 16-11-2009

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

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

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

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

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

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