Reducing the parallel complexity of certain linear programming problems (extended abstract)

Pravin M. Vaidya,
Department of Computer Science,
University of Illinois at Urbana-Champaign,
Urbana, IL 61801.

Abstract. We study the parallel complexity of solving linear programming problems in the context of interior point methods. Let n and m respectively denote the number of variables and the number of constraints in the given problem. We give an algorithm that solves linear programming problems in $O((mn)^{1/4}(\log m)^3 L)$ time using $O(M(n)m/n + n^3)$ processors. ($M(n)$ is the number of operations for multiplying two $n \times n$ matrices.) This gives an improvement in the parallel running time for $n = o(m)$. A typical case where $n = o(m)$ is the dual of the uncapacitated transportation problem. Our algorithm solves the uncapacitated transportation problem in $O((VE)^{1/4}(\log V)^3(\log V\gamma))$ time using $O(V^3)$ processors where V (E) is the number of nodes (edges) and γ is the largest magnitude of an edge cost or a demand at a node. As a byproduct we also obtain a better parallel algorithm for the assignment problem for graphs of moderate density.

1. Introduction

We study the parallel complexity of the linear programming problem

$$\max\ c^T x$$

$$\text{s.t. } Ax \geq b$$

where $x \in \mathbf{R}^n$, $A \in \mathbf{R}^{m \times n}$, and $b \in \mathbf{R}^m$. We assume that the polytope $P = \{x : Ax \geq b\}$ is full dimensional and bounded and that all input numbers are integers. Note that since the polytope is bounded, $m > n$. Let L be defined as

$$L = \log_2(\det_{\max}) + \log_2(c^T c + b^T b) + \log_2(m+n)$$

where $\det_{\max}$ denotes the largest absolute value of the determinant of any square submatrix of A.

We shall study the parallelizability of linear programming in the context of interior point methods. We shall assume a CRCW PRAM model of computation. All the known interior point algorithms for solving linear programming problems are iterative algorithms [2,10,13,15,18]; each iteration typically consists of computations such as linear system solve, matrix inversion, matrix-vector multiplications, and the computations in an iteration are easily implemented in polylogarithmic time given an adequate (polynomial) number of processors. (In other words, the computations during an iteration can be performed in NC.) So the parallel complexity of such algorithms is essentially determined by the number of iterations. In all implementations of interior point algorithms for linear programming it has been observed that the number of iterations grows very slowly with m and n i.e. roughly speaking the number of iterations grow like $O(\log m\, L)$ [1,8,9,16]. On the other hand the best known bound on the number of iterations is $O(\sqrt{m}\, L)$[15]. So it is interesting to study by how much the number of iterations can be reduced. In this paper we shall give an interior point algorithm that solves the above linear programming problem in $O((mn)^{1/4}L)$ iterations; the computations during an iteration can be performed in $O((\log m)^3)$ time using $O(\frac{M(n)m}{n} + n^3)$ processors where $M(n)$ is the number of operations for multiplying two $n \times n$ matrices [5,11]. Thus we obtain an improvement in the number iterations and a faster parallel algorithm for $n = o(m)$.

A typical case where $n = o(m)$ is the dual of the uncapacitated transportation problem [see 14]; here $n = V$ and $m = E$ where V and E are the number of nodes and the number of edges in the given network respectively. Our algorithm solves the uncapacitated transportation problem in $O((VE)^{1/4}(\log V)^3 \log(V\gamma))$ time using $O(V^3)$ processors where γ is the largest magnitude of an edge cost or a demand at a node. As a byproduct we also obtain a faster parallel algorithm for the assignment problem (which is a special case of the transportation problem) for graphs of moderate density. The previously best known algorithm for the assignment problem runs in $O(\min\{E^{1/2}, V^{2/3}\}(\log V)^3 \log(V\gamma))$ time and uses $O(V^3)$ processors [6]; so we obtain an improvement for $E = o(V^{5/3})$.

In section 2 we shall give an overview and in section 3 we shall describe our algorithm. If we think of the problem $\{\max c^T x \mid Ax \geq b\}$ as the dual problem then the corresponding primal problem is $\{\min -b^T \pi \mid -A^T \pi = c,\ \pi \geq 0\}$, $\pi \in \mathbf{R}^m$. In section 4

CH2925-6/90/0000/0583$01.00 © 1990 IEEE

we shall discuss how to construct in parallel a solution to the primal problem from a solution to the dual problem obtained by our algorithm.

2. An overview

Let P be the polytope defined as

$$P = \{x : Ax \geq b\}.$$

For a function $f(x)$ we shall let $\nabla f(x)$ and $\nabla^2 f(x)$ respectively denote the gradient and the Hessian of $f(x)$ evaluated at x. Also, let $\lambda(f, x)$ be defined as

$$\lambda(f, x) = \nabla f(x)^T \nabla^2 f(x)^{-1} \nabla f(x),$$

and let $\Lambda(f)$ be defined as

$$\Lambda(f) = \max_{x \in P} \{\lambda(f, x)\}.$$

Since our algorithm will use Newton's method for minimizing some suitable strictly convex function $f(x)$ at every iteration, we shall briefly review a variant of Newton's Method for minimizing $f(x)$.

Newton's method for minimizing a strictly convex function $f(x)$. At each step in the method we move from the current point z to a point where $f(x)$ has a strictly smaller value. During a step we reset z as

$$z \leftarrow z - t\,B(z)^{-1} \nabla f(z).$$

where $B(z)$ is a good approximation to $\nabla^2 f(z)$ and t is a suitable choice of step length. Variants and details may be found in [12]. ■

A generic path following algorithm. A path following algorithm for solving the linear programming problem $\{\max c^T x \mid Ax \geq b\}$ may be designed as follows. Let $g(x)$ be a continuously twice differentiable function that is strictly convex over the interior of P and that tends to ∞ as we approach any of the boundaries of P. ($g(x)$ could be thought as a smooth convex barrier function.) Then by the Implicit Function Theorem [3,4] it follows that the equation

$$\nabla g(x) = \beta c, \quad \beta \in \mathbf{R}$$

implicitly defines x as function of β, and that this implicit function is a continuously differentiable function of β. As β is changed continuously from $-\infty$ to ∞, $x = x(\beta)$ sweeps a continuous trajectory in the polytope P; the two limit points of this trajectory are the points that minimize and maximize the linear function $c^T x$ over P. The path following algorithm follows the trajectory $\nabla g(x) = t\,c$ to the point that maximizes $c^T x$ over P using (some variant of) Newton's method.

Typically, the path following algorithm generates a strictly increasing sequence of parameters $\beta^0, \beta^1, \cdots, \beta^k, \cdots$ and a sequence of points $x^0, x^1, \cdots, x^k, \cdots$ such that x^k is a good approximation to ω^k where ω^k satisfies the condition

$$\nabla g(\omega^k) = \beta^k c.$$

Note that ω^k is the minimizer of the function $g^k(x)$ over the polytope P where

$$g^k(x) = g(x) - \beta^k c^T x.$$

During the kth iteration we move from x^{k-1} to x^k using some variant of Newton's method for minimizing $g^k(x)$. The algorithm halts when β becomes suitably large, typically $2^{O(L)}$. The number of iterations depends on the rate at which the β's can be increased.

How fast can the β's be increased. The sequence of β's has to be chosen such that x^k can be computed from x^{k-1} in a few ($O(1)$) steps of Newton's method; this limits the rate at which the β's can be increased. In other words β^{k-1} and β^k have to be such that ω^{k-1} and x^{k-1} are close enough to ω^k so that Newton's method for minimizing $g^k(x)$ converges rapidly. If the quantity $\lambda(g^k, x^{k-1})$ is small then

$$\frac{1}{2}\lambda(g^k, x^{k-1}) \approx g^k(\omega^k) - g^k(x^{k-1})$$

and Newton's method converges quickly. Let us assume that if $\lambda(g^k, x^{k-1})$ satisfies the condition

$$\lambda(g^k, x^{k-1}) \leq \delta(g),$$

then x^k is computable from x^{k-1} in $O(1)$ steps of Newton's method. (Here $\delta(g)$ is a suitably small parameter dependent on g.)

We can set up a condition in terms of $\Lambda(g)$ and $\delta(g)$ which tells us how fast the parameter β can be increased. For simplicity let us assume that $x^{k-1} = \omega^{k-1}$. With a little bit of manipulation it is easily seen that

$$\lambda(g^k,\omega^{k-1}) = \left(\frac{\beta^k-\beta^{k-1}}{\beta^{k-1}}\right)^2 \lambda(g,\omega^{k-1})$$
$$\leq \left(\frac{\beta^k-\beta^{k-1}}{\beta^{k-1}}\right)^2 \Lambda(g) .$$

So if we choose β^k such that

$$\frac{\beta^k-\beta^{k-1}}{\beta^{k-1}} \leq \left(\frac{\delta(g)}{\Lambda(g)}\right)^{1/2} .$$

then $\lambda(g^k,\omega^{k-1})$ will be guaranteed to be at most $\delta(g)$ and we will be in the domain of rapid convergence for Newton's method. Rewriting the above condition we get that

$$\beta^k \leq \left(1 + \frac{\delta(g)^{1/2}}{\Lambda(g)^{1/2}}\right)\beta^{k-1} \qquad \cdots (2.1)$$

Note that if β^k violates condition (2.1) there would no apriori guarantee that Newton's method for minimizing $g^k(x)$ would converge rapidly.

A path of analytic centers. In the path following algorithms for linear programming studied thus far [13,15,17,18] the function g has been the logarithmic barrier function for P. The logarithmic barrier function for P, denoted $\phi(x)$, is given by

$$\phi(x) = -\sum_{i=1}^{m} \ln(a_i^T x - b_i)$$

where a_i^T denotes the *ith* row of A. Specifically, these algorithms follow the path of analytic centers [13,15,17] defined as

$$\nabla\phi(x) = \beta c , \quad \beta \in \mathbf{R} .$$

For the logarithmic barrier $\phi(x)$, $\Lambda(\phi)$ could be as large as m and the best value known for $\delta(\phi)$ is $O(1)$. So if (2.1) is to be satisfied and β^k is to be made as large as possible then $\beta^k \approx (1+\frac{1}{m^{1/2}})\beta^{k-1}$, and as a result the bound obtained on the number of iterations is $O(m^{1/2}L)$ [15].

A path of volumetric centers. Another possible choice for the function g in a path following algorithm is the determinant barrier $\frac{1}{2}\ln(\det(\nabla^2\phi(x)))$ where det(.) denotes the determinant. The determinant barrier is strictly convex over P and has been used in [19] to obtain better algorithms for convex programming. A path following algorithm based on the determinant barrier would follow the path of volumetric centers defined by

$$\nabla(\frac{1}{2}\ln(\det(\nabla^2\phi(x)))) = \beta c , \quad \beta \in \mathbf{R} .$$

(The volumetric center is discussed in [19].) Even though the parameter Λ in this case is at most n, the parameter δ is small, about $\frac{1}{m^{1/2}}$[19]. In order to make β^k large without violating (2.1), we have to let $\beta^k \approx (1+\frac{1}{m^{1/4}n^{1/2}})\beta^{k-1}$, and we get a bound of $O(m^{1/4}n^{1/2}L)$ iterations. This bound is better than the $O(\sqrt{m}\, L)$ bound for the path of analytic centers for $m > n^2$ but worse for $m \leq n^2$.

A path of hybrid centers. The parameter δ has a smaller value in the case of the determinant barrier than in the case of the logarithmic barrier because the determinant barrier has loosely speaking a higher degree of non-linearity than the logarithmic barrier. So we add a small multiple of the logarithmic barrier to the determinant barrier to obtain a ***hybrid barrier***; this damps out some of the non-linearity in the determinant and considerably improves the value of δ at the cost of a modest increase in the value of Λ. Let $\psi(x)$ be the hybrid barrier function defined as

$$\psi(x) = \frac{1}{2}\ln(\det(\nabla^2\phi(x))) + \frac{n}{m}\phi(x)$$

and let the path of hybrid centers be defined as

$$\nabla\psi(x) = \beta c , \quad \beta \in \mathbf{R} .$$

In our algorithm g will be the function ψ and we will follow the path of hybrid centers. It can be shown that $\Lambda(\psi) \leq 2n$ and $\delta(\psi) = \Omega(\frac{n^{1/2}}{m^{1/2}})$. So we can choose β^k as $\beta^k \approx (1+\frac{1}{(mn)^{1/4}})\beta^{k-1}$ and still satisfy (2.1). This leads to a bound of $O((mn)^{1/4}L)$ iterations for an algorithm that follows the path of hybrid centers.

The question of finding a barrier function with a better combination of Λ and δ remains open.

3. An algorithm that follows a path of hybrid centers

We shall briefly describe an algorithm for solving linear programming problems that follows a path of hybrid centers. Let $\psi(x)$ be the hybrid barrier function defined as

$$\psi(x) = \frac{1}{2}\ln(\det(\nabla^2\phi(x))) + \frac{n}{m}\phi(x)$$

where $\phi(x) = -\sum_{i=1}^{m}\ln(a_i^Tx - b_i)$, det(.) denotes the determinant, and a_i^T denotes the *ith* row of A. Let $\psi^k(x)$ be defined as

$$\psi^k(x) = \psi(x) - 2n\ln(c^Tx - \beta^k) .$$

and let ω^k denote the minimizer of $\psi^k(x)$ over P^k where

$$P^k = \{x : Ax \geq b\ ,\ c^Tx \geq \beta^k\} .$$

We will call ω^k the hybrid center of P^k. (Note that the definition of ψ^k is slightly different from the one for g^k in section 2; the two definitions correspond to different parametrizations of the same trajectory.) Let $\beta^{\max}$ be the maximum value of the objective function c^Tx over the feasible region (polytope) $P = \{x : Ax \geq b\}$.

The algorithm will generate a strictly increasing sequence of parameters $\beta^0, \beta^1, \cdots, \beta^k, \cdots$ whose limit is $\beta^{\max}$, together with a sequence of points $x^0, x^1, \cdots, x^k, \cdots$ such that x^k is a good approximation to ω^k; specifically,

$$\psi^k(x^k) - \psi^k(\omega^k) \leq \frac{\epsilon n^{1/2}}{m^{1/2}}$$

for some suitably small constant $\epsilon < 1$. We start with a β^0 such that $\beta^{\max} - \beta^0 \leq 2^{O(L)}$, and a good approximation x^0 to ω^0. The issue of obtaining a suitable starting point is addressed in [15,18]. Let $B^k(x)$ be defined as

$$B^k(x) = \sum_{i=1}^{m}(\sigma_i(x) + \frac{n}{m})\frac{a_i a_i^T}{(a_i^Tx - b_i)^2} + \frac{2n}{(c^Tx - \beta^k)^2}cc^T$$

where $\sigma_i(x) = \dfrac{a_i^T\nabla^2\phi(x)^{-1}a_i}{(a_i^Tx - b_i)^2}$, $1 \leq i \leq m$ and $\nabla^2\phi(x) = \sum_{i=1}^{m}\dfrac{a_i a_i^T}{(a_i^Tx - b_i)^2}$. $B^k(x)$ is a good approximation to $\nabla^2\psi^k(x)$ and is used in our algorithm for minimizing $\psi^k(x)$ during the *kth* iteration. Specifically, we have that

$$\forall\, \xi \in \mathbf{R}^n,\ \ \xi^TB^k(x)\xi \leq \xi^T\nabla^2\psi^k(x)\xi \leq 5\,\xi^TB^k(x)\xi .$$

Let $\alpha < 1$ be a small positive constant.

At the beginning of the *kth* iteration we have a parameter β^{k-1}, a point x^{k-1} such that $c^Tx^{k-1} \geq \beta^{k-1}$ and

$$\psi^{k-1}(x^{k-1}) - \psi^{k-1}(\omega^{k-1}) \leq \frac{\epsilon n^{1/2}}{m^{1/2}} .$$

During the *kth* iteration we execute the following steps in sequence.

1. $\beta^k \leftarrow \beta^{k-1} + \dfrac{\alpha}{(mn)^{1/4}}(c^Tx^{k-1} - \beta^{k-1})$.
2. Compute x^k from x^{k-1} by executing $O(1)$ steps of Newton's method for minimizing $\psi^k(x)$ as follows.

```
$z \leftarrow x^{k-1}$;
For $j=1$ to $6\log_2(1/\alpha\epsilon)$ do
    $z \leftarrow z - 0.2\,B^k(z)^{-1}\nabla\psi^k(z)$;
$x^k \leftarrow z$.
```

Bounding the number of iterations. The algorithm halts when $c^Tx^k - \beta^k$ falls below $2^{-\gamma L}$ where $\gamma > 1$ is some positive constant, and an exact optimum is then found as described in [15]. (The computing of an exact optimum involves a projection computation and is easily performed in NC.) If for some constant θ we can show that

$$c^Tx^k - \beta^k \geq \theta(\beta^{\max} - \beta^k)$$

then it will follow that

$$\beta^{\max} - \beta^k \leq (1 - \frac{\theta\alpha}{(mn)^{1/4}})(\beta^{\max} - \beta^{k-1})$$

which will mean that the algorithm halts in $O((mn)^{1/4}L)$ iterations. We shall sketch how the desired lower bound on $c^Tx^k - \beta^k$ is obtained. $\nabla\psi^k(x)$ may be expressed as

$$-\nabla\psi^k(x) = \sum_{i=1}^{m}(\sigma_i(x)+\frac{n}{m})\frac{a_i}{a_i^Tx-b_i} + 2n\frac{c}{c^Tx-\beta^k}$$

and since $\nabla\psi^k(\omega^k)=0$,

$$\sum_{i=1}^{m}(\sigma_i(\omega^k)+\frac{n}{m})\frac{a_i}{a_i^T\omega^k-b_i} + 2n\frac{c}{c^T\omega^k-\beta^k} = 0 .$$

Taking the dot product of both sides of the above equation with $x-\omega^k$, noting that $\sum_{i=1}^{m}\sigma_i(x)=n$, and adding $4n$ to both sides we get that

$$\sum_{i=1}^{m}(\sigma_i(\omega^k)+\frac{n}{m})\frac{a_i^Tx-b_i}{a_i^T\omega^k-b_i} + 2n\frac{c^Tx-\beta^k}{c^T\omega^k-\beta^k} = 4n .$$

Then with a little bit of manipulation it is seen that $\beta^{\max}-\beta^k \leq 2(c^T\omega^k-\beta^k)$, and since x^k and ω^k are close, $c^Tx^k-\beta^k \approx c^T\omega^k-\beta^k$. The desired bound on $c^Tx^k-\beta^k$ then follows. At this point it is worth noting that the method for bounding the number of iterations is similar to the one for the path of analytic centers in [15]; we obtain a better bound because in our case it suffices to put a weight of $2n$ on the objective function plane rather than a weight of m that is used in [15].

The rate at which the β's are increased guarantees that we stay in the domain of rapid convergence for Newton's method, and $O(1)$ steps of Newton's method for minimizing $\psi^k(x)$ suffice to move from x^{k-1} to x^k. Details will appear in the full paper; the techniques are an extension of the ones in [19]. Also, it is straightforward to show that the quantities $\sigma_i(x)$, $B^k(x)$, $B^k(x)^{-1}$, and $\nabla\psi^k(x)$ can be evaluated in $O((\log m)^3)$ time using $O(\frac{M(n)m}{n}+n^3)$ processors; thus a step of Newton's method and an iteration may be excuted in $O((\log m)^3)$ time using the same number of processors.

4. Solving the primal problem

We shall discuss how to solve the primal problem

$$\min -b^T\pi$$

$$\text{s.t. } -A^T\pi = c$$

$$\pi \geq 0$$

where $\pi\in\mathbf{R}^m$, $b\in\mathbf{R}^m$, $A\in\mathbf{R}^{m\times n}$, and $c\in\mathbf{R}^n$. The dual of the above problem is the problem

$$\max c^Tx$$

$$\text{s.t. } Ax\geq b$$

whose solution has been discussed in the previous sections. To solve the primal problem we first solve the dual using the algorithm in section 3, and then construct an almost optimal feasible solution to the primal using the output of the algorithm. The output of the algorithm in section 3 is a point $\tilde{x}$ such that

$$\nabla\psi(\tilde{x}) = -\sum_{i=1}^{m}\frac{\tilde{w}_i a_i}{a_i^T\tilde{x}-b_i} = \tilde{\beta}c + \mathrm{v}$$

where $\tilde{w}_i = \sigma_i(\tilde{x}) + \frac{n}{m}$, $\tilde{\beta}=2^{O(L)}$ and v is a very small error (vector) term. For simplicity we shall assume that $\mathrm{v}=0$; the procedure described below can be modified in a straightforward manner to handle the (small) error term v.

The form of the gradient $\nabla\psi(x)$ leads to a natural correspondence between primal variables and dual slacks for points on the trajectory of hybrid centers, and using this correspondence we will construct an almost optimal solution $\tilde{\pi}$ to the primal problem from $\tilde{x}$. Such a correspondence was first given for the trajectory of analytic centers in [2]. Define $\tilde{\pi}$ as $\tilde{\pi}_i = \frac{\tilde{w}_i}{\tilde{\beta}(a_i^T\tilde{x}-b_i)}$, $1\leq i\leq m$. Then we have that

$$-A^T\tilde{\pi} = c .$$

Thus $\tilde{\pi}$ is feasible for the primal problem since $\tilde{\pi}_i\geq 0$, $1\leq i\leq m$. It can be shown that if π^{opt} is an optimal solution to the primal problem then

$$b^T(\pi^{opt}-\tilde{\pi}) \leq \frac{4n}{\tilde{\beta}} .$$

Thus $\tilde{\pi}$ is almost optimal, and an optimal solution to the primal problem may be constructed by a projection computation as described in [15].

Note that if the error v mentioned above is not zero then $\tilde{\pi}$ has a small component outside the affine space $\{\pi : -A^T\pi = c\}$; specifically,

$-A^T\tilde{\pi} = c + \frac{1}{\tilde{\beta}}\mathbf{v}$. We can construct a small perturbation $\mathbf{u}$ such that $\tilde{\pi}+\mathbf{u}$ is in this affine space; in particular choosing $\mathbf{u} = \frac{1}{\tilde{\beta}}A(A^TA)^{-1}\mathbf{v}$ suffices. If $\mathbf{v}$ is adequately small (its norm $2^{-O(L)}$) then $\tilde{\pi}+\mathbf{u}$ is feasible for the primal problem and is almost optimal. The entire process of constructing $\tilde{\pi}+\mathbf{u}$ from $\tilde{x}$ may be carried out in $O((\log m)^3)$ time using $O(\frac{M(n)m}{n} + n^3)$ processors.

5. References

(1) I. Adler, N. K. Karmarkar, M. G. C. Resende, and G. Veiga, An implementation of Karmarkar's algorithm for linear programming, *Mathematical Programming*, Vol. 44, 1989.

(2) D. A. Bayer and J. C. Lagarias, The non-linear geometry of linear programming I. Affine and Projective scaling trajectories, *Trans. Amer. Math. Soc.*, to appear.

(3) G. A. Bliss, Lectures on the calculus of variations, Phoenix Science Series, The University of Chicago Press, Chicago 37, 1946.

(4) S. Bochner, and W. T Martin, Several Complex variables, Princeton University Press, Princeton, 1948.

(5) D. Coppersmith and S. Winograd, Matrix multiplication via arithmetic progressions, *Proc.* 19*th Annual ACM Symp. Theory of Computing*, (May 1987) 1-6.

(6) A. Goldberg, S. Plotkin, D. Shmoys, and E. Tardos, Interior-point methods in parallel computation, *Proc.* 30*th Annual IEEE Symp. on Foundations of Computer Science*, 1989, pp. 350-355.

(7) M. Grotschel, L. Lovasz, and A. Schrijver, Geometric algorithms and combinatorial optimization, Springer-Verlag Berlin Heidelberg, 1988.

(8) N. K. Karmarkar and K. G. Ramakrishnan, Further developments in the new polynomial time algorithm for linear programming, Talk at the 12*th* International Symposium Mathematical Programming, Boston, Aug. 1985.

(9) N. K. Karmarkar and K. G. Ramakrishnan, Implementation and computational results of Karmarkar's algorithm for linear programming using an iterative method for computing projections, Technical Memorandum, Mathematical Sciences Center, AT&T Bell Laboratories, Murray Hill, NJ, Nov. 1989.

(10) N. Karmarkar, A new polynomial time algorithm for linear programming, Combinatorica, Vol. 4, 1984, pp. 373-395.

(11) V. Pan and J. Reif, Efficient parallel solution of linear systems, *Proc.* 17*th Annual ACM Symposium on Theory of Computing*, 1985, pp. 143-152.

(12) M. Minoux, Mathematical Programming: Theory and Algorithms, John Wiley & Sons Ltd., New York, 1986.

(13) N. Meggido, Progress in Mathematical Programming: Interior Point and related methods (Ed. N. Meggido), Springer-Verlag New York Inc., 1989.

(14) C. H. Papadimitriou and K. Steiglitz, Combinatorial optimization: Algorithms and complexity, Prentice-Hall Inc., Englewood Cliffs, NJ, 1982.

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

(16) D. Shanno and C. Monma, Computational experience with the primal-dual method, Talk at ORSA/TIMS conference, Washington D.C., April 1988..IP 7.

(17) Gy. Sonnevend, An analytical center for polyhedrons and new classes of global algorithms for linear (smooth, convex) functions, Technical Memorandum, Dept. of Numerical Analysis, Institute of Mathematics, Eotvos University, Budapest, Hungary.

(18) P. M. Vaidya, An algorithm for linear programming that requires $O(((m+n)n^2+(m+n)^{1.5}n)L)$ arithmetic operations, *Proceedings* 19*th Annual ACM Symposium Theory of Computing*, (May

1987), pp. 29–38; also to appear in *Mathematical Programming*.

(19) P. M. Vaidya, A new algorithm for minimizing convex functions over convex sets, *Proceedings 30th Annual IEEE Symposium Foundations of Computer Science*, 1989, pp. 338–343.