Practical Optimization Algorithms and Applications

Chapter X: Linear Programming

Lingfeng NIU

Research Center on Fictitious Economy & Data Science,
University of Chinese Academy of Sciences

niulf@ucas.ac.cn

Outline

Outline

Standard Form

Problem

$$\min c^T x \text{ s.t. } Ax = b, x \geq 0, \tag{1}$$

is called the *standard form* of linear programming, where c and x are vectors in $\Re^n$, b is a vector in $\Re^m$, and A is an $m \times n$ matrix. For the standard formulation (15), we will assume throughout that $m < n$.

Standard Form

Problem

$$\min c^T x \text{ s.t. } Ax = b, x \geq 0, \tag{1}$$

is called the *standard form* of linear programming, where c and x are vectors in $\Re^n$, b is a vector in $\Re^m$, and A is an $m \times n$ matrix. For the standard formulation (15), we will assume throughout that $m < n$.

- The linear program (15) is *infeasible* if the feasible set is empty;
- The problem (15) is *unbounded* if the objective functions is unbounded below on the feasible region.

Exercise

Transform

$$\max c^T x \text{ s.t. } Ax \leq b$$

to the standard form.

Exercise

Transform

$$\max c^T x \text{ s.t. } Ax \leq b$$

to the standard form.

- Convert a "maximize" objective into the "minimize" form;
- Add slack variables & surplus variables;
- Split unbounded variables x into nonnegative and nonpositive parts

$$x = x^+ - x^-, \text{ where } x^+ = \max(x, 0) \geq 0 \text{ and } x^- = \max(-x, 0) \geq 0.$$

Exercise

Transform

$$\max c^T x \text{ s.t. } Ax \le b$$

to the standard form.

- Convert a "maximize" objective into the "minimize" form;
- Add slack variables & surplus variables;
- Split unbounded variables x into nonnegative and nonpositive parts

$$x = x^+ - x^-, \text{ where } x^+ = \max(x, 0) \ge 0 \text{ and } x^- = \max(-x, 0) \ge 0.$$

Convert the following linear program to standard form:

$$\begin{aligned} \max_{x,y} \quad & c^T x + d^T y \\ s.t. \quad & A_1 x = b_1, \\ & A_2 x + B_2 y \le b_2, \\ & l \le y \le u. \end{aligned}$$

Outline

Optimality Conditions

The Lagrangian function for (15) can be written as

Optimality Conditions

The Lagrangian function for (15) can be written as

$$\mathcal{L}(x, \lambda, s) = c^T x - \lambda^T (Ax - b) - s^T x. \tag{2}$$

The KKT condition for (15) is

Optimality Conditions

The Lagrangian function for (15) can be written as

$$\mathcal{L}(x, \lambda, s) = c^T x - \lambda^T (Ax - b) - s^T x. \tag{2}$$

The KKT condition for (15) is

$$\begin{aligned} A^T\lambda + s &= c, && \text{(3a)} \\ Ax &= b, && \text{(3b)} \\ x &\geq 0, && \text{(3c)} \\ s &\geq 0, && \text{(3d)} \\ x^T s &= 0. && \text{(3e)} \end{aligned}$$

Convexity of the LP problem ensures that the KKT conditions are sufficient for a global minimum.

The Dual Problem

We call problem

$$\max b^T\lambda, \text{ s.t. } A^T\lambda \le c. \tag{4}$$

the *dual problem* for (15). In contrast, (15) is often referred to as the *primal*. We can restate (4) by introducing a vector of *dual slack variables*, and writing

$$\max b^T\lambda, \text{ s.t. } A^T\lambda + s = c, \; s \ge 0. \tag{5}$$

The variables (λ, s) in this problem are sometimes jointly referred to collectively as *dual variables*.

The primal-dual relationship is symmetric; by taking the dual of the dual problem (4), we recover the primal problem (15).

Strong Duality

Theorem

(i) If either problem (15) or (4) has a solution with finite optimal objective value, then so does the other, and the objective values are equal.
(ii) If either problem (15) or (4) has an unbounded objective, then the other problem has no feasible points.

Outline

Bases and Basic Feasible Points

Assume that the matrix A in (15) has full row rank. A vector x is a *basic feasible point* if it is feasible and there exist a subset $\mathcal{B}$ of the index set $\{1, 2, \cdots, n\}$ such that

- $\mathcal{B}$ contains exactly m indices;
- $i \notin \mathcal{B} \Rightarrow x_i = 0$
- The $m \times m$ matrix defined by $B = [A_i]_{i \in \mathcal{B}}$ is nonsingular, where A_i is the ith column of A.

A set $\mathcal{B}$ satisfying these properties is call a *basis* for the problem (15). The corresponding matrix B is called the *basis matrix*.

Theorem

(i) If there is a feasible point for (15), then there is a basic feasible point.
(ii) If (15) has solutions, then at least one such solution is a basic optimal point.
(iii) If (15) is feasible and bounded, then it has an optimal solution.

Vertices of the Feasible Polytope

The feasible set defined by the linear constraints is a polytope, and the vertices of this polytope are the points that do not lie on a straight line between two other points in the set.

- Geometrically, they are easily recognizable.
- Algebraically, the vertices are exactly the basic feasible points.

We therefore have an important relationship between the algebraic and geometric viewpoints

Theorem

All basic feasible points for (15) are vertices of the feasible polytope $\{x|Ax = b, x \geq 0\}$*, and vice versa.*

Outline of the Simplex Method

All iterates of the simplex method are basic feasible points for (15) and therefore vertices of the feasible polytope. Most steps consist of a move from one vertex to an adjacent one for which the set of basis $\mathcal{B}$ differs in exactly one component.

- On most steps (but not all), the value of the primal objective function $c^T x$ is decreased.
- Another type of step occurs when the problem is unbounded: The step is an edge along which the objective function is reduced, and along which we can move infinitely far without ever reaching a vertex.

The major issue at each simplex iteration is to decide which index to change in the basis $\mathcal{B}$. Unless the step is a direction of unboundedness, one index must be removed from $\mathcal{B}$ and replaced by another from outside $\mathcal{B}$.

Outline: Pricing

From $\mathcal{B}$ and KKT conditions (17), we can derive values for not just the primal variable x but also the dual variables (λ, s).
First, define index set $\mathcal{N}$ as the complement of $\mathcal{B}$,that is, $\mathcal{N} = \{1, 2, \cdots, n\}/\mathcal{B}$.
Just as B is the column submatrix of A that corresponds to the indices $i \in \mathcal{B}$, we use N to denote the submatrix $N = [A_i]_{i \in \mathcal{N}}$. We also partition the n-element vectors x, s, and c according to the index sets $\mathcal{B}$ and $\mathcal{N}$, using the notation

$$\begin{aligned} x_B &= [x_i]_{i \in \mathcal{B}}, \quad x_N = [x_i]_{i \in \mathcal{N}}, \\ s_B &= [s_i]_{i \in \mathcal{B}}, \quad s_N = [s_i]_{i \in \mathcal{N}}, \\ c_B &= [c_i]_{i \in \mathcal{B}}, \quad c_N = [c]_{i \in \mathcal{N}}. \end{aligned}$$

From the KKT condition (17b), we have that

$$Ax = Bx_B + Nx_N = b.$$

The primal variable x for this simplex iterate is defined as

$$x_B = B^{-1} b \geq 0, \qquad x_N = 0. \tag{6}$$

Outline: Pricing

We choose s to satisfy the complementarity condition (17c) by setting $s_B = 0$. The remaining components λ and s_N can be found by partitioning condition (17a) into c_B and c_N components and using $s_B = 0$ to obtain

$$B^T \lambda = c_B, \qquad N^T \lambda + s_N = c_N. \tag{7}$$

Since B is square and nonsingular, nonsingular, the first equation uniquely defines λ as

$$\lambda = B^{-T} c_B. \tag{8}$$

The second equation implies a value for s_N:

$$s_N = c_N - N^T \lambda = c_N - (B^{-1} N)^T c_B. \tag{9}$$

Computation of the vector s_N is often referred to as *pricing*. The components of s_N are often called the *reduced costs* of the nonbasic variables x_N.

Outline: Pivoting

The only KKT condition that we have not enforced explicitly is the nonnegativity condition $s \geq 0$. The basic components s_B certainly satisfy this condition, by our choice $s_B = 0$.

- If the vector s_N defined by (9) also satisfies $s_N \geq 0$, we have found an optimal vector triple (x, λ, s), so the algorithm can terminate and declare success.
- The usual case, however, is that one or more of the components of s_N are negative, so the condition $s \geq 0$ is violated. The new index to enter the basic $\mathcal{B}$ - *the entering index* - is chosen to be one of the indices $q \in \mathcal{N}$ for which $s_q < 0$.

Outline: Pivoting

Our procedure for altering $\mathcal{B}$ and changing x and s accordingly is as follows:

- allow x_q to increase from zero during the next step;
- fix all other components of x_N at zero;
- figure out the effect of increasing x_q on the current basic vector x_B, given that we want to stay feasible with respect to the equality constraints $Ax = b$;
- keep increasing x_q until one of the components of x_B (x_p, say) is driven to zero, or determining that no such component exists (the unbounded case);
- remove index p (known as the *leaving index*) from $\mathcal{B}$ and replace it with the entering index q.

The process of selecting entering and leaving indices, and performing the algebraic operations necessary to keep track of the values of the variables x, λ and s, is known as *pivoting*.

Outline: Pivoting

Since both the new iterate x^+ and the current iterate x should satisfy $Ax = b$, and since $x_N = 0$ and $x_i^+ = 0$ for $i \in \mathcal{N} \setminus \{q\}$, we have

$$Ax^+ = Bx_B^+ + A_q x_q^+ = Bx_B = Ax.$$

By multiplying this expression by B^{-1} and rearranging, we obtain

$$x_B^+ = x_B - B^{-1} A_q x_q^+. \tag{10}$$

Geometrically speaking, (10) is a move along an edge of the feasible polytope that decreases $c^T x$. We continue to move along this edge until a new vertex is encountered. We have to stop at this vertex, since by definition we cannot move any further without leaving the feasible region. At the new vertex, a new constraint $x_i \geq 0$ must have become active, that is, one of the components x_p, $p \in \mathcal{B}$, has decreased to zero. We then remove this index p from the basis $\mathcal{B}$ and replace it by q.

Outline: Finite Termination

We now show how the step defined by (10) affects the value of $c^T x$. From (10), we have

$$c^T x^+ = c_B^T x_B^+ + c_q x_q^+ = c_B^T x_B - c_B^T B^{-1} A_q x_q^+ + c_q x_q^+. \tag{11}$$

From (8), we have $c_b^T B^{-1} = \lambda^T$. Sine $q \in \mathcal{N}$, from the second equation in (7) we have $A_q^T \lambda = c_q - s_q$. Therefore,

$$c_B^T B^{-1} A_q x_q^+ = \lambda^T A_q x_q^+ = (c_q - s_q) x_q^+,$$

so by substituting in (11) we obtian

$$c^T x^+ = c_B^T x_B - (c_q - s_q) x_q^+ + c_q x_q^+ = c^T x + s_q x_q^+.$$

Since q was chosen to have $s_q < 0$, it follows that the step (10) produces a decrease in the primal objective function $c^T x$ whenever $x_q^+ > 0$.

Outline: Finite Termination

- It is possible that we can increase x_q^+ to ∞ without ever encountering a new vertex. In other words, the constraint $x_B^+ = x_B - B^{-1}A_q x_q^+ \geq 0$ holds for all positive values of x_q^+. When this happens, the linear program is *unbounded*; the simplex method has identified a ray that lies entirely within the feasible polytope along which the objective $c^T x$ decreases to $-\infty$.
- If for any basis $\mathcal{B}$, $\forall i \in \mathcal{B}$, $x_i \neq 0$, then we are guaranteed that $x_q^+ > 0$, so we can be assured of a strict decrease in the objective function $c^T x$ at this step. Furthermore, if the linear program (15) is also bounded, the simplex method terminates at a basic optimal point.

A Single Step of the Method

```
Given $\mathcal{B}, \mathcal{N}, x_B = B^{-1}b \geq 0, x_N = 0$;
    Solve $B^T\lambda = c_B$ for $\lambda$,
    Compute $s_N = c_N - N^T\lambda$;
    **if** $s_N \geq 0$
        **STOP**; (* optimal point found *)
    Select $q \in \mathcal{N}$ with $s_q < 0$ as the entering index;
    Solve $Bt = A_q$ for $t$;
    **if** $t \leq 0$
        **STOP**; (* problem is unbounded *)
    Calculate $x_q^+ = \min_{i \mid t_i > 0} (x_B)_i / t_i$, and use $p$ to denote the index of the basic
        variable for which this minimum is achieved;
    Update $x_B^+ = x_B - t x_q^+, x_N^+ = (0, \ldots, 0, x_q^+, 0, \ldots, 0)^T$;
    Change $\mathcal{B}$ by adding $q$ and removing $p$.
```

Exercise

Start with the basis $\mathcal{B} = \{3, 4\}$, use the simplex method solve the following linear programming:

$$\begin{aligned} \min \quad & -3x_1 - 2x_2 \\ s.t. \quad & x_1 + x_2 + x_3 = 5, \\ & 2x_1 + \tfrac{1}{2}x_2 + x_4 = 8, \\ & x \geq 0. \end{aligned}$$

Exercise

Start with the basis $\mathcal{B} = \{3, 4\}$, use the simplex method solve the following linear programming:

$$\begin{aligned} \min \quad & -3x_1 - 2x_2 \\ s.t. \quad & x_1 + x_2 + x_3 = 5, \\ & 2x_1 + \tfrac{1}{2}x_2 + x_4 = 8, \\ & x \geq 0. \end{aligned}$$

$$x^* = [11/3, 4/3, 0, 0]^T$$

Other Important Details

- Pricing and Selection of the Entering Index
- Starting the Simplex Method
- Degenerate Steps and Cycling
- Presolving

Pricing and Selection of the Entering Index

There are usually many negative components of s_N at each step. We need to select one of these, s_q, as the entering variable. How do we make this selection?

- Ideally, we would like to choose the sequence of entering variables that gets us to the solution x^* in the fewest possible steps, but we rarely have the global perspective needed to implement this strategy.
- Instead, we use more shortsighted but practical strategies that obtain a significant decrease in $c^T x$ on just the present iteration.
- There is usually a tradeoff between the effort spent searching for a good entering variable q that achieves this aim and the decrease in $c^T x$ to be obtained from pivoting on q. Different pivot strategies resolve the tradeoff in different ways.

Starting the Simplex Method

The simplex method requires a basic feasible starting point x and a corresponding initial basic index set $B \subset \{1, 2, \cdots, n\}$ with $|\mathcal{B}| = m$ such that

- the basis matrix B is nonsingular;
- $x_B = B^{-1}b \geq 0$ and $x_N = 0$.

The problem of finding this initial point and basis may itself be nontrivial - in fact, its difficulty is equivalent to that of actually finding the solution of a linear program.

Starting the Simplex Method

One approach for dealing with this difficulty is the *two-phase approach*.

- In Phase I, we set up a linear program different from (15), whose initial basic feasible point is trivial to determine and solution gives a basic feasible initial point for Phase II. Solve this auxiliary problem with the simplex method.
- In Phase II, a second linear program very similar to the original problem (15) is solved, starting from the Phase-I solution. The solution of the original problem (15) can be extracted easily from the solution of the Phase-II problem.

Two-Phase Approach - Phase I

In Phase I we introduce artificial variables z into (15) and redefine the objective function to be the sum of these artificial variables. To be specific, the Phase-I problem is

$$\min e^T z, \text{ s.t. } Ax + Ez = b, (x, z) \geq 0, \tag{12}$$

where $z \in \Re^m$, $e = (1, 1, \cdots, 1)^T$, and E is a diagonal matrix whose diagonal elements are

$$E_{jj} = +1 \text{ if } b_j \geq 0, \qquad E_{jj} = -1 \text{ if } b_j < 0.$$

It is easy to see that the point (x, z) defined by

$$x = 0, \qquad z_j = |b_j|, j = 1, 2, \cdots, , m, \tag{13}$$

is a basic feasible point for (12). Obviously, this point satisfies the constraints in (12), while the initial basis matrix B is simply the matrix E, which is clearly nonsingular.

Two-Phase Approach - Phase I

At any feasible point for (12), the artificial variables z represent the amounts by which the constraints $Ax = b$ are violated by the x component. The objective function is simply the sum of these violations, so by minimizing this sum we are forcing x to become feasible for the original problem (15). In fact, it is not difficult to see that the Phase-I problem (12) has an optimal objective value of zero if and only if the original problem (15) has feasible points by using the following argument:

- If there exists a vector $(\hat{x}, \hat{z})$ that is feasible for (12) such that $e^T \hat{z} = 0$, we must have $\hat{z} = 0$, and therefore $A\hat{x} = b$ and $\hat{x} \geq 0$, so $\hat{x}$ is feasible for the original problem (15).
- Conversely, if $\hat{x}$ is feasible for (15), then the point $(\hat{x}, 0)$ is feasible for (12) with an objective value of 0. Since the objective in (12) is obviously nonnegative at all feasible points, then $(\hat{x}, 0)$ must be optimal for (12), verifying our claim.

Two-Phase Approach - Phase II

We can now apply the simplex method to (12) from the initial point (13). This linear program cannot be unbounded, because its objective function is bounded below by 0, so the simplex method will terminate at an optimal point.

- If the objective $e^T z$ is positive at this solution, we conclude by the argument above that the original problem (15) is infeasible.
- Otherwise, the simplex method identifies a point $(\hat{x}, \hat{z})$ with $e^T \hat{z} = 0$, which is a basic feasible point for the following *Phase-II problem*:

$$\min c^T x \text{ s.t. } Ax + z = b, x \geq 0, 0 \geq z \geq 0. \tag{14}$$

Two-Phase Approach - Phase II

Note that phase II problem (14) is equivalent to (15), because any solution (and indeed any feasible point) must have $z = 0$. We need to retain the artificial variables z in Phase II, however, since some components of z may be present in the optimal basis from Phase I that we are using as the initial basis for (12), though of course the values $\hat{z}_j$ of these components must be zero. In fact, we can modify (14) to include only those components of z that are present in the optimal basis for (14), omitting the others.

Presolving

Presolving (also know as preprocessing) is carried out in practical linear programming codes to reduce the size of the user-defined linear programming problem before passing it the solver.

- The simplest preprocessing check is for the presence of *zero rows and columns*.
- The *row singleton* happens when one of the inequality constraints involves just one of the variables.
- The *free column singleton* happens when there is a variable that occurs in only of the equality constraints and is free.
- An other presolving technique is to check for *forcing or dominated constraints*.

Presolving techniques are applied recursively, because the elimination of certain variables or constraints may create situations that allow further eliminations.

Reference: E.D. Andersen and K.D. Andersen, *Presolving in linear programming.* Mathematical Programming, 71(1995), pp. 221-245.

Exercise

Consider the following linear program

$$\begin{aligned} \min \quad & -5x_1 - x_2 \\ s.t. \quad & x_1 + x_2 \le 5, \\ & 2x_1 + \tfrac{1}{2}x_2 \le 8, \\ & x \ge 0. \end{aligned}$$

(a) Add slack variables x_3 and x_4 to convert this problem to standard form.
(b) Solve this problem using the simplex method, showing at each step the basis and the vector λ, s_N, and x_B, and the value of the objective function.

The Dual Simplex Method

The method introduced above starts with a feasible x (with $x_B \geq 0$ and $x_N = 0$) and a corresponding dual iterates (λ, s) for which $s_B = 0$ but s_N is not necessarily nonnegative. After making systematic column interchanges between B and N, it finally reaches a feasible dual point (λ, s) at which $s_N \geq 0$, thus yielding a solution of both the primal problem (15) and the dual (4).

The dual simplex method starts with a point (λ, s) at which $s_N \geq 0$ and $s_B = 0$, and a corresponding primal feasible point x for which $x_N = 0$ but x_B is not necessarily nonnegative. By making systematic column interchanges between B and N, it finally reaches a feasible primal point x for which $x_B \geq 0$, signifying optimality.

Where dose the Simplex Method fit?

In linear programming, as in all optimization problems in which inequality constraints (including bounds) are present, the fundamental issue is to partition the inequality constraints into those that are *active* at the solution and those that are *inactive*. The simplex method belongs to a general class of algorithms for constrained optimization known as *active set methods*, which explicitly maintain estimates of the active and inactive index sets that are updated at each step of the algorithm. (In the case of simplex and linear programming, $\mathcal{B}$ is the set of "probably inactive" indices - those for which the bound $x_i \geq 0$ is inactive¡ªwhile $\mathcal{N}$ is the set of "probably active" indices.) Like most active set methods, the simplex method makes only modest changes to these index sets at each step: A single index is transferred from $\mathcal{B}$ into $\mathcal{N}$, and vice versa.

Where dose the Simplex Method fit?

- The simplex method is highly efficient on almost all practical problems (the method generally requires at most $2m$ to $3m$ iterations, where m is the row dimension of the constraint matrix in (15)).
- There are pathological problems on which the algorithm performs very poorly. Klee and Minty presented an n-dimensional problem whose feasible polytope has $2n$ vertices, for which the simplex method visits every single vertex before reaching the optimal point!

The complexity of the simplex method is exponential; roughly speaking, its running time can be an exponential function of the dimension of the problem.

Where dose the Simplex Method fit?

For many years, theoreticians searched for a linear programming algorithm that has polynomial complexity, that is, an algorithm in which the running time is bounded by a polynomial function of the size of the input.

- In the late 1970s, Khachiyan described an *ellipsoid method* that indeed has polynomial complexity but turned out to be much too slow in practice.
- In the mid-1980s, Karmarkar described a polynomial algorithm that approaches the solution through the interior of the feasible polytope rather than working its way around the boundary as the simplex method does. Karmarkar's announcement marked the start of intense research in the field of interior-point methods, which are the subject of the next section.

Outline

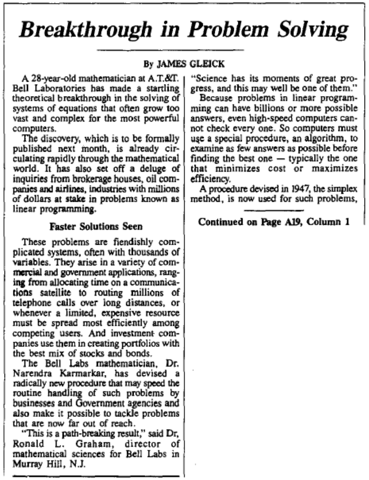
Introduction

Roughly speaking, the time required to solve a linear program may be exponential in the size of the problem, as measured by the number of unknowns and the amount of storage needed for the problem data. For almost all practical problems, the simplex method is much more efficient than this bound would suggest, but its poor worst-case complexity motivated the development of new algorithms with better guaranteed performance.

The first such method was the ellipsoid method, proposed by Khachiyan(A polynomial algorithm in linear programming, Soviet Mathematics Doklady,20 (1979), pp. 191–194), which finds a solution in time that is at worst polynomial in the problem size. Unfortunately, this method approaches its worst-case bound on all problems and is not competitive with the simplex method in practice.

Karmarkar's projective algorithm(new polynomial-time algorithm for linear programming, Combinatorics, 4(1984), pp. 373–395), announced in 1984, also has the polynomial complexity property, but it came with the added attraction of good practical behavior.

Introduction



Breakthrough in Problem Solving

By JAMES GLEICK

A 28-year-old mathematician at A.T.&T. Bell Laboratories has made a startling theoretical breakthrough in the solving of systems of equations that often grow too vast and complex for the most powerful computers.

The discovery, which is to be formally published next month, is already circulating rapidly through the mathematical world. It has also set off a deluge of inquiries from brokerage houses, oil companies and airlines, industries with millions of dollars at stake in problems known as linear programming.

Faster Solutions Seen

These problems are fiendishly complicated systems, often with thousands of variables. They arise in a variety of commercial and government applications, ranging from allocating time on a communications satellite to routing millions of telephone calls over long distances, or whenever a limited, expensive resource must be spread most efficiently among competing users. And investment companies use them in creating portfolios with the best mix of stocks and bonds.

The Bell Labs mathematician, Dr. Narendra Karmarkar, has devised a radically new procedure that may speed the routine handling of such problems by businesses and Government agencies and also make it possible to tackle problems that are now far out of reach.

"This is a path-breaking result," said Dr. Ronald L. Graham, director of mathematical sciences for Bell Labs in Murray Hill, N.J.

"Science has its moments of great progress, and this may well be one of them."

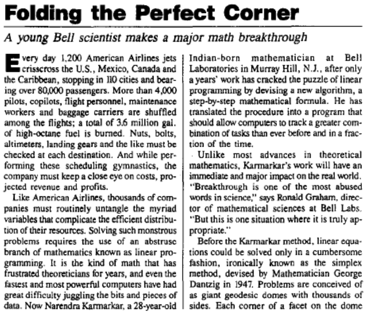
Because problems in linear programming can have billions or more possible answers, even high-speed computers cannot check every one. So computers must use a special procedure, an algorithm, to examine as few answers as possible before finding the best one — typically the one that minimizes cost or maximizes efficiency.

A procedure devised in 1947, the simplex method, is now used for such problems,

Continued on Page A19, Column 1

THE NEW YORK TIMES, November 19, 1984

Karmarkar at Bell Labs: an equation to find a new way through the maze



Folding the Perfect Corner

A young Bell scientist makes a major math breakthrough

Every day 1,200 American Airlines jets crisscross the U.S., Mexico, Canada and the Caribbean, stopping in 110 cities and bearing over 80,000 passengers. More than 4,000 pilots, copilots, flight personnel, maintenance workers and baggage carriers are shuffled among the flights; a total of 3.6 million gal. of high-octane fuel is burned. Nuts, bolts, altimeters, landing gears and the like must be checked at each destination. And while performing these scheduling gymnastics, the company must keep a close eye on costs, projected revenue and profits.

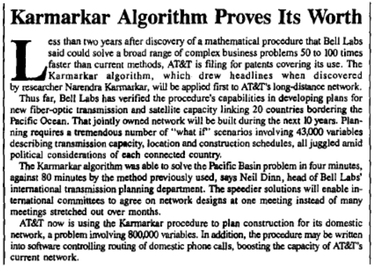
Like American Airlines, thousands of companies must routinely untangle the myriad variables that complicate the efficient distribution of their resources. Solving such monstrous problems requires the use of an abstruse branch of mathematics known as linear programming. It is the kind of math that has frustrated theoreticians for years, and even the fastest and most powerful computers have had great difficulty juggling the bits and pieces of data. Now Narendra Karmarkar, a 28-year-old Indian-born mathematician at Bell Laboratories in Murray Hill, N.J., after only a years' work has cracked the puzzle of linear programming by devising a new algorithm, a step-by-step mathematical formula. He has translated the procedure into a program that should allow computers to track a greater combination of tasks than ever before and in a fraction of the time.

Unlike most advances in theoretical mathematics, Karmarkar's work will have an immediate and major impact on the real world. "Breakthrough is one of the most abused words in science," says Ronald Graham, director of mathematical sciences at Bell Labs. "But this is one situation where it is truly appropriate."

Before the Karmarkar method, linear equations could be solved only in a cumbersome fashion, ironically known as the simplex method, devised by Mathematician George Dantzig in 1947. Problems are conceived of as giant geodesic domes with thousands of sides. Each corner of a facet on the dome

TIME MAGAZINE, December 3, 1984

Introduction



Karmarkar Algorithm Proves Its Worth

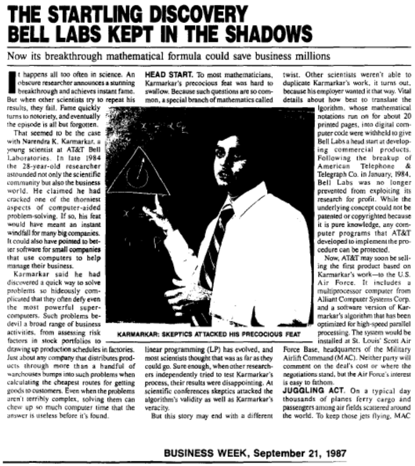
Less than two years after discovery of a mathematical procedure that Bell Labs said could solve a broad range of complex business problems 50 to 100 times faster than current methods, AT&T is filing for patents covering its use. The Karmarkar algorithm, which drew headlines when discovered by researcher Narendra Karmarkar, will be applied first to AT&T's long-distance network.

Thus far, Bell Labs has verified the procedure's capabilities in developing plans for new fiber-optic transmission and satellite capacity linking 20 countries bordering the Pacific Ocean. That jointly owned network will be built during the next 10 years. Planning requires a tremendous number of "what if" scenarios involving 43,000 variables describing transmission capacity, location and construction schedules, all juggled amid political considerations of each connected country.

The Karmarkar algorithm was able to solve the Pacific Basin problem in four minutes, against 80 minutes by the method previously used, says Neil Dinn, head of Bell Labs' international transmission planning department. The speedier solutions will enable international committees to agree on network designs at one meeting instead of many meetings stretched out over months.

AT&T now is using the Karmarkar procedure to plan construction for its domestic network, a problem involving 800,000 variables. In addition, the procedure may be written into software controlling routing of domestic phone calls, boosting the capacity of AT&T's current network.

THE WALL STREET JOURNAL, July 18, 1986



THE STARTLING DISCOVERY BELL LABS KEPT IN THE SHADOWS

Now its breakthrough mathematical formula could save business millions

It happens all too often in science. An obscure researcher announces a stunning breakthrough and achieves instant fame. But when other scientists try to repeat his results, they fail. Fame quickly turns to notoriety, and eventually the episode is all but forgotten.

That seemed to be the case with Narendra K. Karmarkar, a young scientist at AT&T Bell Laboratories. In late 1984 the 28-year-old researcher astounded not only the scientific community but also the business world. He claimed he had cracked one of the thorniest aspects of computer-aided problem-solving. If so, his feat would have meant an instant windfall for many big companies. It could also have pointed to better software for small companies that use computers to help manage their business.

Karmarkar said he had discovered a quick way to solve problems so hideously complicated that they often defy even the most powerful supercomputers. Such problems bedevil a broad range of business activities, from assessing risk factors in stock portfolios to drawing up production schedules in factories. Just about any company that distributes products through more than a handful of warehouses bumps into such problems when calculating the cheapest routes for getting goods to customers. Even when the problems aren't terribly complex, solving them can chew up so much computer time that the answer is useless before it's found.

HEAD START. To most mathematicians, Karmarkar's precocious feat was hard to swallow. Because such questions are so common, a special branch of mathematics called linear programming (LP) has evolved, and most scientists thought that was as far as they could go. Sure enough, when other researchers independently tried to test Karmarkar's process, their results were disappointing. At scientific conferences skeptics attacked the algorithm's validity as well as Karmarkar's veracity.

But this story may end with a different twist. Other scientists weren't able to duplicate Karmarkar's work, it turns out, because his employer wanted it that way. Vital details about how best to translate the lgorithm, whose mathematical notations run on for about 20 printed pages, into digital computer code were withheld to give Bell Labs a head start at developing commercial products. Following the breakup of American Telephone & Telegraph Co. in January, 1984, Bell Labs was no longer prevented from exploiting its research for profit. While the underlying concept could not be patented or copyrighted because it is pure knowledge, any computer programs that AT&T developed to implement the procedure can be protected.

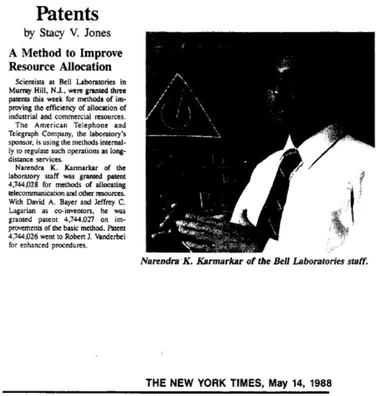
Now, AT&T may soon be selling the first product based on Karmarkar's work—to the U.S. Air Force. It includes a multiprocessor computer from Alliant Computer Systems Corp. and a software version of Karmarkar's algorithm that has been optimized for high-speed parallel processing. The system would be installed at St. Louis' Scott Air Force Base, headquarters of the Military Airlift Command (MAC). Neither party will comment on the deal's cost or where the negotiations stand, but the Air Force's interest is easy to fathom.

JUGGLING ACT. On a typical day thousands of planes ferry cargo and passengers among air fields scattered around the world. To keep those jets flying, MAC

KARMARKAR: SKEPTICS ATTACKED HIS PRECOCIOUS FEAT

BUSINESS WEEK, September 21, 1987

Introduction



Patents

by Stacy V. Jones

A Method to Improve Resource Allocation

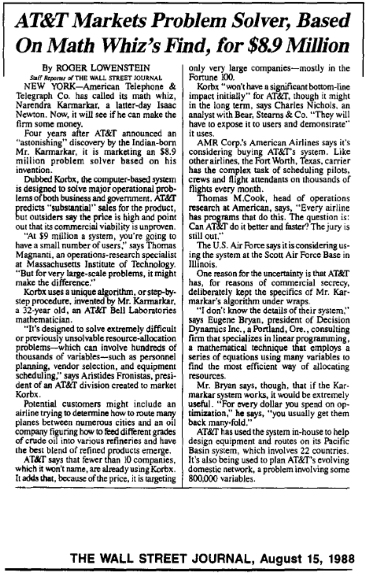
Scientists at Bell Laboratories in Murray Hill, N.J., were granted three patents this week for methods of improving the efficiency of allocation of industrial and commercial resources.

The American Telephone and Telegraph Company, the laboratory's sponsor, is using the methods internally to regulate such operations as long-distance services.

Narendra K. Karmarkar of the laboratory staff was granted patent 4,744,028 for methods of allocating telecommunication and other resources. With David A. Bayer and Jeffrey C. Lagarian as co-inventors, he was granted patent 4,744,027 on improvements of the basic method. Patent 4,744,026 went to Robert J. Vanderbei for enhanced procedures.

Narendra K. Karmarkar of the Bell Laboratories staff.

THE NEW YORK TIMES, May 14, 1988



AT&T Markets Problem Solver, Based On Math Whiz's Find, for $8.9 Million

By ROGER LOWENSTEIN

Staff Reporter of THE WALL STREET JOURNAL

NEW YORK—American Telephone & Telegraph Co. has called its math whiz, Narendra Karmarkar, a latter-day Isaac Newton. Now, it will see if he can make the firm some money.

Four years after AT&T announced an "astonishing" discovery by the Indian-born Mr. Karmarkar, it is marketing an $8.9 million problem solver based on his invention.

Dubbed Korbx, the computer-based system is designed to solve major operational problems of both business and government. AT&T predicts "substantial" sales for the product, but outsiders say the price is high and point out that its commercial viability is unproven.

"At $9 million a system, you're going to have a small number of users," says Thomas Magnanti, an operations-research specialist at Massachusetts Institute of Technology. "But for very large-scale problems, it might make the difference."

Korbx uses a unique algorithm, or step-by-step procedure, invented by Mr. Karmarkar, a 32-year old, an AT&T Bell Laboratories mathematician.

"It's designed to solve extremely difficult or previously unsolvable resource-allocation problems—which can involve hundreds of thousands of variables—such as personnel planning, vendor selection, and equipment scheduling," says Aristides Fronistas, president of an AT&T division created to market Korbx.

Potential customers might include an airline trying to determine how to route many planes between numerous cities and an oil company figuring how to feed different grades of crude oil into various refineries and have the best blend of refined products emerge.

AT&T says that fewer than 10 companies, which it won't name, are already using Korbx. It adds that, because of the price, it is targeting only very large companies—mostly in the Fortune 100.

Korbx "won't have a significant bottom-line impact initially" for AT&T, though it might in the long term, says Charles Nichols, an analyst with Bear, Stearns & Co. "They will have to expose it to users and demonstrate" it uses.

AMR Corp.'s American Airlines says it's considering buying AT&T's system. Like other airlines, the Fort Worth, Texas, carrier has the complex task of scheduling pilots, crews and flight attendants on thousands of flights every month.

Thomas M.Cook, head of operations research at American, says, "Every airline has programs that do this. The question is: Can AT&T do it better and faster? The jury is still out."

The U.S. Air Force says it is considering using the system at the Scott Air Force Base in Illinois.

One reason for the uncertainty is that AT&T has, for reasons of commercial secrecy, deliberately kept the specifics of Mr. Karmarkar's algorithm under wraps.

"I don't know the details of their system," says Eugene Bryan, president of Decision Dynamics Inc., a Portland, Ore., consulting firm that specializes in linear programming, a mathematical technique that employs a series of equations using many variables to find the most efficient way of allocating resources.

Mr. Bryan says, though, that if the Karmarkar system works, it would be extremely useful. "For every dollar you spend on optimization," he says, "you usually get them back many-fold."

AT&T has used the system in-house to help design equipment and routes on its Pacific Basin system, which involves 22 countries. It's also being used to plan AT&T's evolving domestic network, a problem involving some 800,000 variables.

THE WALL STREET JOURNAL, August 15, 1988

Primal-Dual Methods

Consider the linear programming problem in standard form

$$\min c^T x \text{ s.t. } Ax = b, x \geq 0. \tag{15}$$

The dual problem for this problem is

$$\max b^T \lambda, \text{ s.t. } A^T \lambda + s = c, \ s \geq 0. \tag{16}$$

The solution of these two problems are characterized by the following KKT conditions

$$\begin{aligned} A^T \lambda + s &= c, && \text{(17a)} \\ Ax &= b, && \text{(17b)} \\ x^T s &= 0, && \text{(17c)} \\ x &\geq 0, && \text{(17d)} \\ s &\geq 0. && \text{(17e)} \end{aligned}$$

Primal-Dual Methods

Primal-dual methods find solutions (x^*, λ^*, s^*) of the KKT system by applying variants of Newton's method to the three equalities and modifying the search directions and step lengths so that the inequalities $(x, s) \geq 0$ are satisfied *strictly* at every iteration.

- The equations are only mildly nonlinear and so are not difficult to solve by themselves.
- The problem becomes much more difficult when we add the nonnegativity requirement. The nonnegativity condition is the source of all the complications in the design and analysis of interior-point methods.

Primal-Dual Methods

To derive primal-dual interior-point methods, we restate the KKT optimality conditions in a slightly different form by means of a mapping F from $\Re^{2n+m}$ to $\Re^{2n+m}$:

$$\begin{aligned} F(x,\lambda,s) = \begin{bmatrix} A^T\lambda + s - c \\ Ax - b \\ XSe \end{bmatrix} &= 0, && (18a) \\ (x,s) &\geq 0, && (18b) \end{aligned}$$

where

$$X = \mathrm{diag}(x_1, x_2, \cdots, x_n), \qquad S = \mathrm{diag}(s_1, s_2, \cdots, s_n), \tag{19}$$

and $e = (1, 1, \cdots, 1)^T \in \Re^n$. Primal-dual methods generate iterates (x_k, λ_k, s_k) that satisfy the bounds (18b) strictly, that is, $x_k > 0$ and $s_k > 0$. This property is the origin of the term *interior-point*.

Primal-Dual Methods

Like most iterative algorithms in optimization, primal-dual interior-point methods have two basic ingredients:

- **a procedure for determining the step**: The procedure for determining the search direction has its origins in Newton's method for the nonlinear equation (18a);
- **a measure of the desirability of each point in the search space**; An important component of measure of desirability is the average value of the pairwise products $x_i s_i$, $i = 1, 2, \cdots, n$, which are positive when $x > 0$ and $s > 0$. This quantity is known as the *duality measure* and is defined as follows:

$$\mu = \frac{1}{n} \sum_{i=1}^{n} x_i s_i = \frac{x^T s}{n}. \tag{20}$$

Primal-Dual Methods: The Search Direction

Consider Newton's method for the nonlinear equations (18a):

$$J(x, \lambda, s) \begin{bmatrix} \triangle x \\ \triangle \lambda \\ \triangle s \end{bmatrix} = -F(x, \lambda, s), \tag{21}$$

where J is the Jacobian of F, if we use the notation r_c and r_b for the first two block rows in F, that is

$$r_b = Ax - b. \qquad r_c = A^T\lambda + s - c, \tag{22}$$

we can write the Newton equation (21) as follows:

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \triangle x \\ \triangle \lambda \\ \triangle s \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ -XSe \end{bmatrix}. \tag{23}$$

Primal-Dual Methods: The Search Direction

Usually, a full step along this direction is not permissible, since it would violate the bound $(x, s) \geq 0$. To avoid this difficulty, we perform a line search along the Newton direction so that the new iterate is

$$(x, \lambda, s) + \alpha(\triangle x, \triangle \lambda, \triangle s), \tag{24}$$

for some line search parameter $\alpha \in (0, 1]$.

Unfortunately, we often can take only a small step along the direction $(\alpha \ll 1)$ before violating the condition $(x, s) > 0$; hence, the pure Newton direction, which is known as the *affine scaling direction*, often does not allow us to make much progress toward a solution.

Primal-Dual Methods: The Search Direction

Most primal-dual methods use a less aggressive Newton direction, one that does not aim directly for a solution of (18a) but rather for a point whose pairwise products $x_i s_i$ are reduced to a lower average value - not all the way to zero. Specifically, we take a Newton step towards a point for which $x_i s_i = \sigma\mu$, where μ is the current duality measure and $\sigma \in [0, 1]$ is the reduction factor that we wish to achieve in the duality measure on this step. The modified step equation is then

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \triangle x \\ \triangle \lambda \\ \triangle s \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ -XSe + \sigma\mu e \end{bmatrix}. \tag{25}$$

We call σ the *centering parameter*.

Primal-Dual Path-Following Framework

```
Given $(x^0, \lambda^0, s^0) \in \mathcal{F}^o$
for $k = 0, 1, 2, \ldots$
    Solve
        $$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S^k & 0 & X^k \end{bmatrix} \begin{bmatrix} \Delta x^k \\ \Delta \lambda^k \\ \Delta s^k \end{bmatrix} = \begin{bmatrix} -r_c^k \\ -r_b^k \\ -X^k S^k e + \sigma_k \mu_k e \end{bmatrix},$$
        where $\sigma_k \in [0, 1]$ and $\mu_k = (x^k)^T s^k / n$;
    Set
        $$(x^{k+1}, \lambda^{k+1}, s^{k+1}) = (x^k, \lambda^k, s^k) + \alpha_k (\Delta x^k, \Delta \lambda^k, \Delta s^k),$$
        choosing $\alpha_k$ such that $(x^{k+1}, s^{k+1}) > 0$.
end (for).
```

The choices of centering parameter σ_k and step length α_k are crucial to the performance of the method. Techniques for controlling these parameters, directly and indirectly, give rise to a wide variety of methods with diverse properties.

The Central Path

The primal-dual feasible set $\mathcal{F}$ and strictly feasible set $\mathcal{F}^0$ are defined as follows:

$$\mathcal{F} = \{(x, \lambda, s) | Ax = b, A^T\lambda + s = c, (x, s) \geq 0\} \tag{26}$$
$$\mathcal{F}^0 = \{(x, \lambda, s) | Ax = b, A^T\lambda + s = c, (x, s) > 0\} \tag{27}$$

The central path $\mathcal{C}$ is an arc of strictly feasible points that plays a vital role in primal-dual algorithms. It is parameterized by a scalar $\tau > 0$, and each point $(x_\tau, \lambda_\tau, s_\tau) \in \mathcal{C}$ satisfies the following equations:

$$\begin{aligned} A^T\lambda + s &= c, && \text{(28a)} \\ Ax &= b, && \text{(28b)} \\ x_i s_i &= \tau, \qquad i = 1, 2, \cdots, n, && \text{(28c)} \\ (x, s) &> 0. && \text{(28d)} \end{aligned}$$

These conditions differ from the KKT conditions only in the term τ on the right-hand side of (28c). Instead of the complementary condition, we require that the pairwise products $x_i s_i$ have the same (positive) value for all indices i.

The Central Path

From (28), we can define the central path as $\mathcal{C} = \{(x_\tau, \lambda_\tau, s_\tau) | \tau > 0\}$. It can be shown that $(x_\tau, \lambda_\tau, s_\tau)$ is defined uniquely for each $\tau > 0$ if and only if $\mathcal{F}^0$ is nonempty.
The condition (28) are also the optimality conditions for a logarithmic-barrier formulation of the problem (15). By introducing log-barrier terms for the nonnegativity constraints, with barrier parameter $\tau > 0$, we obtain

$$\min c^T x - \tau \sum_{i=1}^{n} \ln x_i, \text{ s.t. } Ax = b. \tag{29}$$

The Central Path

Another way of defining $\mathcal{C}$ is to use the mapping F defined in (18) and write

$$F(x_\tau, \lambda_\tau, s_\tau) = \begin{bmatrix} 0 \\ 0 \\ \tau e \end{bmatrix}, \qquad (x_\tau, s_\tau) > 0. \tag{30}$$

The equations (28) approximate the KKT conditions more and more closely as τ goes to zero. If τ converges to anything as $\tau \downarrow 0$, it must converge to a primal-dual solution of the linear program. The central path thus guides us to a solution along a route that steers clear of spurious solutions by keeping all x and s components strictly positive and decreasing the pairwise products $x_i s_i$, $i = 1, 2, \cdots, n$, to zero at roughly the same rate.

The Central Path

Primal-dual algorithms take Newton steps toward points on $\mathcal{C}$ for which $\tau > 0$, rather than pure Newton steps for F. Since these steps are biased toward the interior of the nonnegative orthant defined by $(x, s) \geq 0$, it usually is possible to take longer steps along them than along the pure Newton (affine scaling) steps for F, before violating the positivity condition.
In the feasible case of $(x, \lambda, s) \in \mathcal{F}$, we have $r_b = 0$ and $r_c = 0$, so the search direction becomes

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \triangle x \\ \triangle \lambda \\ \triangle s \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -XSe + \sigma \mu e \end{bmatrix}, \tag{31}$$

where μ is the duality measure and $\sigma \in [0, 1]$ is the centering parameter. When $\sigma = 1$, the equations (31) define a *centering direction*, a Newton step toward the point $(x_\mu, \lambda_\mu, s_\mu) \in \mathcal{C}$, at which all the pairwise products $x_i s_i$ are identical to μ.

The Central Path

- $\sigma = 1$ Centering directions are usually biased strongly toward the interior of the nonnegative orthant and make little, if any, progress in reducing the duality measure μ.
- $\sigma = 0$ gives the standard Newton step, sometimes known as the affine-scaling direction.

Many algorithms use intermediate values of σ from the open interval $(0, 1)$ to trade off between the twin goals of reducing μ and improving centrality.

Path-Following Methods

Path-following algorithms explicitly restrict the iterates to a neighborhood of the central path $\mathcal{C}$ and follow $\mathcal{C}$ to a solution of the linear program. By preventing the iterates from coming too close to the boundary of the nonnegative orthant, they ensure that search directions calculated from each iterate make at least some minimal amount of progress toward the solution.

A key ingredient of any optimization algorithm is a measure of the desirability of each point in the search space. In path-following algorithms, the duality measure μ fills this role. The duality measure μ_k is forced to zero as $k \to \infty$, so the iterates (x_k, λ_k, s_k) come closer and closer to satisfying the KKT conditions.

Central Path Neighborhoods

The two most interesting neighborhoods of $\mathcal{C}$ are the so-called 2-norm neighborhood $\mathcal{N}_2(\theta)$ defined by

$$\mathcal{N}_2(\theta) = \{(x, \lambda, s) \in \mathcal{F}^0 | \|XSe - \mu e\|_2 \leq \theta\mu\}, \tag{32}$$

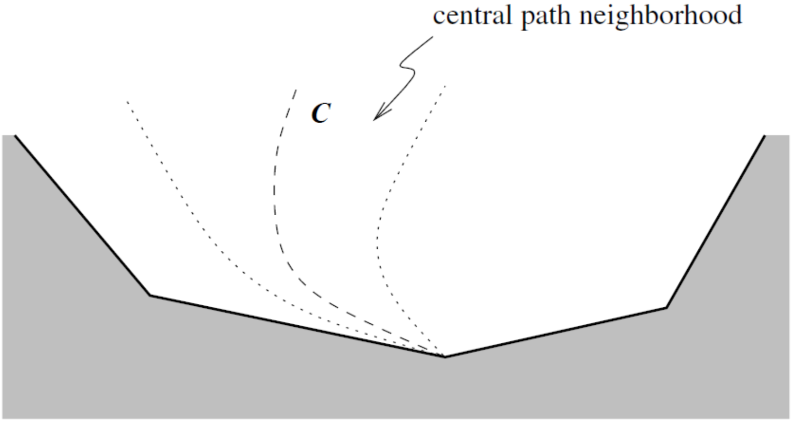
for some $\theta \in [0, 1)$, and the one-sided ∞-norm neighborhood $\mathcal{N}_{\infty}(\gamma)$ defined by

$$\mathcal{N}_{\infty}(\gamma) = \{(x, \lambda, s) \in \mathcal{F}^0 | x_i s_i \geq \gamma\mu, \forall i = 1, 2, \cdots, n\}, \tag{33}$$

for some $\gamma \in (0, 1]$. (Typical values of the parameters are $\theta = 0.5$ and $\gamma = 10^{-3}$.) By keeping all iterates inside one or another of these neighborhoods, path-following methods reduce all the pairwise products $x_i s_i$ to zero at more or less the same rate.

Central Path Neighborhood

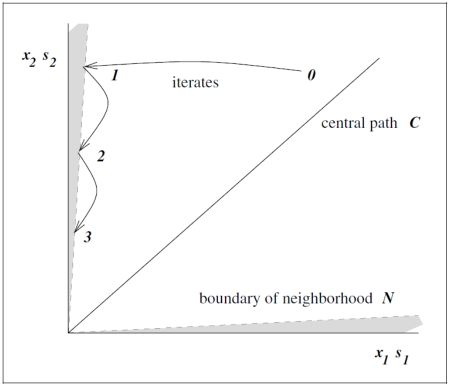
A plot of $\mathcal{C}$ for a typical problem, projected into the space of primal variables x is shown in the following figure:



Long-Step Path-Following Algorithm

```
Given $\gamma$, $\sigma_{min}$, $\sigma_{max}$ with $\gamma \in (0,1)$, $0 < \sigma_{min} \leq \sigma_{max} < 1$, and
$(x^0, \lambda^0, s^0) \in \mathcal{N}_{-\infty}(\gamma)$;
for $k = 0, 1, 2, \cdots$
    Choose $\sigma_k \in [\sigma_{min}, \sigma_{max}]$;
    Solve
        $\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S^k & 0 & X^k \end{bmatrix} \begin{bmatrix} \triangle x^k \\ \triangle \lambda^k \\ \triangle s^k \end{bmatrix} = \begin{bmatrix} -r_c^k \\ -r_b^k \\ -X^k S^k e + \sigma_k \mu_k e \end{bmatrix}$
    to obtain $(\triangle x^k, \triangle \lambda^k, \triangle s^k)$;
    Choose $\alpha_k$ as the largest value of $\alpha$ in $[0,1]$ such that
        $(x^k(\alpha), \lambda^k(\alpha), s^k(\alpha)) \in \mathcal{N}_{-\infty}(\gamma)$;
    Set $(x^{k+1}, \lambda^{k+1}, s^{k+1}) = (x^k(\alpha_k), \lambda^k(\alpha_k), s^k(\alpha_k))$;
end(for).
```

Typical Behavior for $n = 2$.



The horizontal and vertical axes represent $x_1 s_1$ and $x_2 s_2$, so $\mathcal{C}$ is the line emanating from the origin at an angle of $\frac{\pi}{4}$. (A point at the origin is a primal-dual solution if it also satisfies the feasibility conditions.) In the unusual geometry of this figure, the search directions $(\triangle x^k, \triangle \lambda^k, \triangle s^k)$ transform to curves rather than straight lines.

Long-Step Path-Following

As the previous figure shows (and the analysis confirms), the lower bound σ_{min} on the centering parameter ensures that each search direction starts out by moving away from boundary of $\mathcal{N}_{-\infty}(\gamma)$ and into relative interior of this neighborhood.

- Small steps along the search direction improve the centrality.
- Larger values of α take us outside the neighborhood again, since the error in approximating the nonlinear system (30) by the linear step equations (31) becomes more pronounced as α increases.

Still, we are guaranteed that a certain minimum step can be taken before we reach the boundary of $\mathcal{N}_{-\infty}(\gamma)$.

Theorem

Given $\epsilon \in (0, 1)$ and $\gamma \in (0, 1)$, suppose the starting point in the long-step path-following algorithm satisfies $(x^0, \lambda^0, s^0) \in \mathcal{N}_{-\infty}(\gamma)$. Then there is an index K with $K = O(n \log 1/\epsilon)$, such that

$$\mu_k \in \epsilon\mu_0, \quad \text{for all } k \geq K. \tag{34}$$

Practical Primal-Dual Algorithms

- Practical implementations of interior-point algorithms follow the spirit we just discussed, in that strict positivity of x^k and s^k is maintained throughout and each step is a Newton-like step involving a centering component.
- However, most implementations work with an infeasible starting point and infeasible iterations.

Several aspects of "theoretical" algorithms are typically ignored, while several enhancements are added that have a significant effect on practical performance.

Practical Primal-Dual Algorithms

Most existing interior-point codes for general-purpose linear programming problems are based on a predictor-corrector algorithm proposed by Mehrotra. The two key features of this algorithm are

- addition of a *corrector step* to the search direction of primal-dual path-following framework, so that the algorithm more closely follows a trajectory to the primal-dual solution set;
- adaptive choice of the centering parameter σ.

Predicator Step

A key feature of practical algorithms is their use of corrector steps that compensate for the linearization error made by the affine-scaling step in modeling the equation $x_i s_i = 0$, $i = 1, 2, \cdots, n$. Consider the affine-scaling direction $(\triangle x, \triangle\lambda, \triangle s)$ defined by

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \triangle x^{aff} \\ \triangle\lambda^{aff} \\ \triangle s^{aff} \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ -XSe \end{bmatrix}. \tag{35}$$

If we take a full step in this direction, we obtain

$$\begin{aligned} &(x_i + \triangle x_i^{aff})(s_i + \triangle s_i^{aff}) \\ &= x_i s_i + x_i \triangle s_i^{aff} + s_i \triangle x_i^{aff} + \triangle x_i^{aff} \triangle s_i^{aff} \\ &= \triangle x_i^{aff} \triangle s_i^{aff}. \end{aligned}$$

That is, the update value of $x_i s_i$ is $\triangle x_i^{aff} \triangle s_i^{aff}$ rather than the ideal value of 0.

Corrector Step

We can solve the following system to obtain a step $(\triangle x^{cor}, \triangle\lambda^{cor}, \triangle s^{cor})$ that attempts to correct for this deviation from the ideal:

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \triangle x^{cor} \\ \triangle\lambda^{cor} \\ \triangle s^{cor} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -\triangle X^{aff} \triangle S^{aff} e \end{bmatrix}.$$

In many cases, the combined step

$$(\triangle x^{aff}, \triangle\lambda^{aff}, \triangle s^{aff}) + (\triangle x^{cor}, \triangle\lambda^{cor}, \triangle s^{cor})$$

dose a better job of reducing the duality measure than dose the affine-scaling step alone.

Centering Step

Practical algorithms make use of centering steps, with an adaptive choice of the centering parameter σ_k. The affine-scaling step can be used as the basis of a successful heuristic for choosing σ_k. Roughly speaking,

- if the affine-scaling step (multiplying by a steplength to maintain nonnegative of x and s) reduces the duality measure significantly, there is not much need for centering, so a smaller value of σ_k is appropriate.
- Conversely, if not much progress can be made along this direction before reaching the boundary of the nonnegative orthant, a larger value of σ_k will ensure that the next iterate is more centered, so a longer step will be possible from this next point.

Centering Step

Specifically, this scheme calculates the maximum allowable steplengths along the affine-scaling direction (35) as follows:

$$\alpha_{aff}^{pri} \equiv \min(1, \min_{i:\triangle x_i^{aff}<0} -\frac{x_i}{\triangle x_i^{aff}}), \tag{36a}$$

$$\alpha_{aff}^{dual} \equiv \min(1, \min_{i:\triangle s_i^{aff}<0} -\frac{s_i}{\triangle s_i^{aff}}), \tag{36b}$$

and then defines μ_{aff} to be the value of μ that would be obtained by using these steplengths, that is

$$\mu_{aff} = (x + \alpha_{aff}^{pri} \triangle x^{aff})^T (s + \alpha_{aff}^{pri} \triangle s^{aff})/n \tag{37}$$

The centering parameter σ is chosen according to the following heuristic (which does not have a solid analytical justification, but appears to work well in practice):

$$\sigma = (\frac{\mu_{aff}}{\mu})^3.$$

Computations of the Search Direction

Computations of the search direction requires the solution of two linear system.

- First, the system (35) is solved to obtain the affine-scaling direction, also known as the *predictor step*. This step is used to define the right-hand side for the corrector step and to calculate the centering parameter from (36) and (37).
- Second, the search direction is calculated by solving

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \triangle x \\ \triangle \lambda \\ \triangle s \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ -XSe - \triangle X^{aff} \triangle S^{aff} e + \sigma \mu e \end{bmatrix}. \quad (38)$$

  Not that the predictor, corrector, and centering contributions have been aggregated on the right-hand side of this system. The coefficient of the matrix needs to be computed only once, and the marginal cost of solving the second system is relatively small.

Step Lengths

Practical implementations typically do not enforce membership of the central path neighborhoods $\mathcal{N}_2$ and $\mathcal{N}_{-\infty}$. Rather, they calculate the maximum steplengths that can be taken in the x and s variables without violating nonnegativity, then take a steplength of slightly less than this maximum (but no greater than 1).

$$\alpha_{k,max}^{pri} \equiv \min_{i:\triangle x_i^k<0} -\frac{x_i^k}{\triangle x_i^k}, \qquad \alpha_{k,max}^{dual} \equiv \min_{i:\triangle s_i^k<0} -\frac{s_i^k}{\triangle s_i^k}, \tag{39}$$

are the largest values of α for which $x^k + \alpha \triangle x^k \geq 0$ and $s^k + \alpha \triangle s^k \geq 0$, respectively. The following formula is used to calculate steplengths in many practical implementations

$$\alpha_k^{pri} = \min(1, \eta_k \alpha_{k,max}^{pri}), \qquad \alpha_k^{dual} = \min(1, \eta_k \alpha_{k,max}^{dual}), \tag{40}$$

where $\eta_k \in [0.9, 1.0)$ is chosen so that $\eta_k \to 1$ near the solution, to accelerate the asymptotic convergence.

Starting Point

Choice of starting point is an important practical issue with a significant effect on the robustness of the algorithm. We describe here a heuristic that finds a starting point that satisfies the equality constraints in the primal and dual problems reasonable well, while maintaining positivity of the x and s components and avoiding excessively large value of these components.

First, we find a vector $\bar{x}$ of minimum norm satisfying the primal constraint $Ax = b$, and a vector $(\bar{\lambda}, \bar{s})$ satisfy the dual constraints $A^T\lambda + s = c$ such that $\bar{s}$ has minimum norm. That is, we solve the problems

$$\min_{x} x^T x \text{ s.t. } Ax = b,$$

$$\min_{(\lambda,s)} s^T s \text{ s.t. } A^T\lambda + s = c.$$

It is not difficult to show that

$$\bar{x} = A^T(AA^T)^{-1}b, \bar{\lambda} = (AA^T)^{-1}Ac, \bar{s} = c - A^T\bar{\lambda}.$$

Starting Point

In general, $\bar{x}$ and $\bar{s}$ will have nonpositive components, so are not suitable for use as a starting point. We define

$$\delta_x = \max(-(3/2)\min_i \bar{x}_i, 0), \qquad \delta_s = \max(-(3/2)\min_i \bar{s}_i, 0),$$

and adjust the x and x vectors are as follows:

$$\hat{x} = \bar{x} + \delta_x e, \qquad \hat{s} = \bar{s} + \delta_s e,$$

where, $e = (1, 1, \cdots, 1)^T$. Clearly, we have $\hat{x} \geq 0$ and $\hat{s} \geq 0$. To ensure that the components of x^0 and s^0 are not close to zero and not too dissimilar, we add two more scalars defined as follows:

$$\hat{\delta}_x = \frac{1}{2}\frac{\hat{x}^T\hat{s}}{e^T\hat{s}}, \qquad \hat{\delta}_s = \frac{1}{2}\frac{\hat{x}^T\hat{s}}{e^T\hat{x}}.$$

Finally, we define the starting point as follows:

$$x^0 = \hat{x} + \hat{\delta}_x e, \ \ \lambda^0 = \bar{\lambda}, \ \ s^0 = \hat{s} + \hat{\delta}_s e.$$

The computational cost of finding (x^0, λ^0, s^0) by this scheme is about the same as one step of the primal-dual method.

Predictor-Corrector Algorithm(Mehrotra)

```
Calculate $(x^0, \lambda^0, s^0)$ as described above;
for k =0, 1, 2, ...
    Set $(x, \lambda, s) = (x^k, \lambda^k, s^k)$ and solve (35) for $(\Delta x^{aff}, \Delta\lambda^{aff}, \Delta s^{aff})$;
    Calculate $\alpha_{aff}^{pri}$, $\alpha_{aff}^{dual}$ and $\mu_{aff}$ as in (36) and (37);
    Set centering parameter to $\sigma = (\mu_{aff}/\mu)^3$;
    Solve (38) for $(\Delta x, \Delta\lambda, \Delta s)$;
    calculate $\alpha_k^{pri}$ and $\alpha_k^{dual}$ from (40);
```

$$\begin{aligned} x^{k+1} &= x^k + \alpha_k^{pri}\Delta x, \\ (\lambda^{k+1}, s^{k+1}) &= (\lambda^k, s^k) + \alpha_k^{dual}(\Delta\lambda, \Delta s); \end{aligned}$$

```
end(for).
```

Other Primal-Dual Algorithms and Extensions

- Other Path-Following Methods
- Potential-Reduction Methods
- Extensions
  - Monotone Linear Complementary Problem
  - Convex Quadratic Programming
  - Semidefinite Programming

Outline

Conclusion

Interior-point methods share common features that distinguish them from the simplex method.

- The simplex method works its way around the boundary of the feasible polytope, testing a sequence of vertices in turn until it finds the optimal one. Interior-point methods approach the boundary of the feasible set only in the limit. They may approach the solution either from the interior or the exterior of the feasible region, but they never actually lie on the boundary of this region.
- Each interior-point iteration is expensive to compute and can make significant progress towards the solution, while the simplex method usually requires a larger number of inexpensive iterations.

Perspectives

In general,

- simplex codes are faster on problems of small-medium dimensions,
- while interior-point codes are competitive and often faster on large problems.

However, this rule is certainly not hard-and-fast; it depends strongly on the structure of the particular application.

Perspectives

Interior point methods are generally not able to take full advantage of prior knowledge about the solution, such as an estimate of the solution itself or an estimate of the optimal basis. Hence interior-point methods are less usefull than simplex approaches in situations in which “warm-start” information is readily available.

- Branch-and-bound algorithms for solving integer programs , where each node in the branch-and-bound tree requires the solution of a linear program that differs only slightly from one already solved in the parent node.
- Solving a sequence of linear programs in which the data is perturbed slightly to investigate sensitivity of the solutions to various perturbations, or in which we approximate a nonlinear optimization problem by a sequence of linear programs.

Software

Interior-point software has the advantage that it is easy to program, relative to the simplex method. The most complex operation is the solution of the large linear systems at each iteration to compute the step, software to perform this linear algebra operation is readily available.

- LIPSOL is written in entirely in the Matlab language, apart from a small amount of FORTRAN code that interfaces to the linear algebra software.
- PCx is written in C, but also is easy for the interested user to comprehend and modify.

It is even possible for a non-expert in optimization to write an efficient interior-point implementation from scratch that is customized to their particular.

Software

Several state-of-the-art software:

- Simplex methods: CPLEX and XPRESS-MP
- Freely available interior-point codes: PCx, LIPSOL, HOPDM, BPMPD

Thanks for your attention!