Annals of Operations Research
https://doi.org/10.1007/s10479-019-03186-2

S.I.: GAME THEORY AND OPTIMIZATION

A linear programming primer: from Fourier to Karmarkar

Atlanta Chakraborty[1] · Vijay Chandru[2] · M. R. Rao[3]

Abstract
The story of linear programming is one with all the elements of a grand historical drama. The original idea of testing if a polyhedron is non-empty by using a variable elimination to project down one dimension at a time until a tautology emerges dates back to a paper by Fourier in 1823. This gets re-invented in the 1930s by Motzkin. The real interest in linear programming happens during World War II when mathematicians ponder best ways of utilising resources at a time when they are constrained. The problem of optimising a linear function over a set of linear inequalities becomes the focus of the effort. Dantzig's Simplex Method is announced and the Rand Corporation becomes a hot bed of computational mathematics. The range of applications of this modelling approach grows and the powerful machinery of numerical analysis and numerical linear algebra becomes a major driver for the advancement of computing machines. In the 1970s, constructs of theoretical computer science indicate that linear programming may in fact define the frontier of tractable problems that can be solved effectively on large instances. This raised a series of questions and answers: Is the Simplex Method a polynomial-time method and if not can we construct novel polynomial time methods, etc. And that is how the Ellipsoid Method from the Soviet Union and the Interior Point Method from Bell Labs make their way into this story as the heroics of Khachiyan and Karmarkar. We have called this paper a primer on linear programming since it only gives the reader a quick narrative of the grand historical drama. Hopefully it motivates a young reader to delve deeper and add another chapter.

Keywords Linear inequalities · Linear programming · Optimisation · Duality · Simplex method · Ellipsoid method · Interior point method · Monotone

This paper is an extended version of the original paper "The French Connection" by the authors Vijay Chandru and M R Rao as a part of the series "175 Years of Linear Programming" published in Resonance-Journal of Science Education, Vol 3, Issue 10, October 1998, pp 26–40. This paper also draws on some material in "Linear Programming," Chapter 31 by V. Chandru and M.R.Rao in Handbook of Algorithms and Theory of Computing, edited by M.J.Atallah, CRC Press 1999, 31-1 to 31-37.

✉ M. R. Rao
mr_rao@isb.edu

[1] Department of Mathematics, Indian Institute of Science, Bangalore, India

[2] Robert Bosch Centre for Cyber Physical Systems, Indian Institute of Science, Bangalore, India

[3] Indian School of Business, Hyderabad, India

Published online: 01 March 2019

1 Introduction

We are all familiar with computing a solution to simultaneous linear equations by eliminating variables sequentially and followed by back substitution. However, elimination in linear inequalities requires more work and is related to polyhedral theory and linear programming, which we shall discuss in the subsequent sections.

The history of Linear Programming dates back at least as far as 1827 when Joseph Fourier, a celebrated French mathematician, formulated the first linear programming problem as a problem of solving a system of linear inequalities and published a method for solving it.

However, linear programming was developed as a formal discipline only as recently as the 1940s, motivated initially by the need to solve complex planning problems in wartime operations. The World War II provided both the urgency and funds for such research as efficient resource allocation was required for large scale military planning. Its development accelerated rapidly in the postwar period as many industries found valuable uses for linear programming to allocate a finite set of resources in an optimal way.

The founders of the subject are generally regarded as George B. Dantzig and John Von Neumann. Dantzig devised the simplex algorithm in 1947 along with his associates at the US Dept. of Air Force. Dantzig arranged a meeting with John Von Neumann in the same year to discuss his simplex method. Neumann immediately conjectured the theory of LP Duality by realising that the problem he had been working on in game theory was equivalent. Both Dantzig and von Neumann recognised that the linearity assumptions of the model were restrictive but key to efficient solutions.

In 1939, Leonid Kantorovich, a Soviet USSR mathematician, had also developed a similar LP problem with the goal to improve economic planning in USSR. Around the same time, American economist T.C. Koopmans started formulating classic economic problems as linear programs and developed a special linear programming solution used to plan the optimal movement of ships back and forth across the Atlantic during the war. Kantorovich and Koopmans later shared the Nobel Prize in Economics in 1975. Important contributions from the erstwhile Soviet Union was the Ellipsoid Method due first to Shor and specialised for approximate convex optimisation by Nemirovsky and Yudin in 1972. Leonid Khachiyan specialised it for linear optimisation and proved the polynomial-time solvability of linear programming, an important breakthrough in 1979. These historical contributions are detailed in the 1982 paper by Grötschel et al. (1982).

Khachiyan's breakthrough did not prove to be a viable alternative to the Simplex Method as the workhorse of linear programming. It was Narendra Karmarkar, a young computer scientist of Indian origin at AT&T Bell labs, who showed that an interior point method, invariant to projective transformations, could yield both a theoretically efficient and practical method for linear programming.

Linear programming addresses two fundamental problems:

(i) *Solvability* This is the problem of checking if a system of simultaneous linear inequalities on real variables is solvable or not. Geometrically, we have to check if a (convex) polyhedron, defined by such constraints, is non-empty.
(ii) *Optimisation* Linear programming aims to optimise a linear objective function subject to linear equality and inequality constraints.

Fourier's method for solving a system of linear inequalities can be used for solving linear programming problems. But it is not a polynomial time algorithm and moreover not appropriate from a practical point of view for solving large-sized problems, i.e. problems with a large number of inequalities and variables. The Simplex method for linear programming

problems developed by Dantzig is also not a polynomial time algorithm but has been found to be useful, with several bells and whistles added to it over the years, for solving large sized problems. As noted above, it was first the Ellipsoid Method and then the Interior Point Method that turned out to be provably of polynomial-time complexity with the latter also yielding practical algorithms for large scale linear programming. Dantzig's simplex method, with various modifications for speeding up the algorithm, continues to be indispensable for solving linear programming problems and most optimisation platforms today offer a choice of Simplex and Interior-Point Solvers with the latter finding more use in really large scale linear programming problems. Large scale problems have been effectively solved by the primal–dual variants of interior point methods using a "predictor-corrector" approach credited to Kojima, Megiddo and Mizuno. For a readable commentary on the theory and practice of interior point methods, the reader is directed to the exposition by Todd (1994).

2 Fourier's elimination method

Solving a system of the form $Ax \leq b$, where A is a $m \times n$ matrix of real numbers, x is a $n \times 1$ matrix and b is a $m \times 1$ matrix, is the essential task of linear programming. We now describe an elegant syntactic method for this problem known as Fourier elimination.[1] The idea behind it is to derive implied equations/inequalities in which one or more variables are erased. By recursively applying the technique, we arrive at a one or two variable system which can be solved by inspection. The required property is that a solution to the lower dimensional representation can always be extended to a solution of the original system.

Observe that variable elimination in linear inequalities is different from elimination in equations. If we have two equations in two variables, we can take a linear combination and derive a third equation with one of the variables eliminated. The third equation along with the first (or the second) describes the same solution as the original system. This invariance is lost when we move to inequalities and we have to contend with a growing system. Fourier made the remarkable observation that this growth can be contained to be quadratic, per eliminated variable, since it suffices to consider pairs of inequalities which satisfy certain syntactic patterns, as we shall see in the next section. For connections with symbolic reasoning and constraint logic programming, the reader is directed to Lassez and Maher (1988).

2.1 Syntactics

Suppose we wish to eliminate the first variable x_1 from the system $Ax \leq b$. Our goal is to create an equivalent system of linear inequalities $\tilde{A}\tilde{x} \leq \tilde{b}$ defined on the variables $\tilde{x} = (x_2, x_3, \ldots, x_n)$. All inequalities containing x_1 are paired so that the signs of the coefficient of x_1 in each pair are opposite. If x_1 appears in an inequality with a negative sign, it is rewritten so that x_1 appears alone on the right hand side and conversely on the left-hand side if the sign is positive. The inequality constraint set S is thus partitioned into three sets:

T = set of inequalities of S in which x_1 has a negative coefficient
U = set of inequalities of S in which x_1 has a positive coefficient
V = set of inequalities of S which doesn't contain x_1

[1] This method is sometimes referred to as Fourier–Motzkin Elimination in the literature as there was presumably an independent rediscovery (Fourier 1826) of the method over a 100 years later by Motzkin (1936), Dantzig and Eaves (1973) and extended by Cerkinov (1961) and Chandru (1993).

For each pair of inequalities, one belonging to T and the other in U, we get a new inequality with x_1 eliminated. This is the first step to the Fourier elimination. Note that if the sets T and U are roughly half the total number of inequalities of S, the result will be an addition of a quadratic number of new inequalities. Thus, the projection of polyhedra to lower dimensional spaces can create an exponential growth in representation.

We note that both T and U cannot be empty for otherwise x_1 is already eliminated from the system of inequalities. In the case when either T or U is empty, i.e. the variable x_1 always appears with the same sign in S, we need consider only the set V in which x_1 has its coefficient as 0, because the set U can easily be satisfied by an appropriate choice of x_1 given any values for the other variables, which also satisfy the set V. If in addition, V is also empty, the system is *strongly solvable*. A set of inequalities S is said to be strongly solvable if and only if any modification of the right-hand side constants of the inequalities of S results in a new solvable set Lassez (1991). If one of T or U is an empty set along with V, we have a situation that x_1 is monotone positive (or negative) in all inequalities. Hence for any choice of b, we can set x_1 to satisfy all inequalities.

If all the inequalities are deleted in the first step itself, the equivalent system $\tilde{A}\tilde{x} \leq \tilde{b}$ is undefined and we declare the original system strongly solvable. Else, we repeat this construction with $\tilde{A}\tilde{x} \leq \tilde{b}$ to eliminate another $\tilde{x_j}$, $j \neq i$ and so on until all variables are eliminated. After eliminating all n variables, if the final $\tilde{b}$ is non-negative we declare the original inequality system as being solvable otherwise the system is unsolvable.

We shall demonstrate the method with three examples.

Example 1 Let S denote the system of linear inequalities

$$\{-x_1 + 2x_2 \leq 3, 2x_1 - 7x_2 \leq -15, -x_1 \leq 0, -x_2 \leq 0\}.$$

Let us eliminate x_1. We write all the inequalities involving x_1 in the following way:

$$\begin{aligned} x_1 &\geq 2x_2 - 3 \\ x_1 &\leq \frac{7}{2}x_2 - \frac{15}{2} \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

Now the elimination of x_1 is achieved by setting the maximum to be no larger than the minimum. Thus for any $\tilde{x}$ solving $\tilde{A}\tilde{x} \leq \tilde{b}$, it is ensured that we will obtain a non empty range for x_1. Hence, in order to eliminate x_1, we get $2x_2 - 3 \leq \frac{7}{2}x_2 - \frac{15}{2} \implies x_2 \geq 3$. Also $0 \leq \frac{7}{2}x_2 - \frac{15}{2} \implies x_2 \geq \frac{15}{7}$. If we take $x_2 = 5$ (which satisfies both inequalities), after back substituting this value in S, we obtain $7 \leq x_1 \leq 10$ as the corresponding range of feasible values for x_1.

Equivalently S could be written of the form $Ax \leq b$ where

$$A = \begin{pmatrix} -1 & 2 \\ 2 & -7 \\ -1 & 0 \\ 0 & -1 \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} 3 \\ -15 \\ 0 \\ 0 \end{pmatrix}. \tag{1}$$

Elimination of a variable is equivalent to adding a positive multiple of one inequality to another. Hence, eliminating x_1 is equivalent to multiplying A and b on the left by a non-negative matrix R where $R = \begin{pmatrix} 2 & 1 & 0 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$.

Now $RAx \leq Rb \implies -3x_2 \leq -9,\ -7x_2 \leq -15, -x_2 \leq 0$. The sets U and V are empty and hence the system is strongly solvable.

Example 2 As another example let S be given by

$$\{x + 4y - z \leq 2,\ -2x - 3y + z \leq 1,\ -3x - 2y + z \leq 0,\ 4x + y - z \leq -1\}$$

Here,

$$A = \begin{pmatrix} 1 & 4 & -1 \\ -2 & -3 & 1 \\ -3 & -2 & 1 \\ 4 & 1 & -1 \end{pmatrix}, \quad u = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} 2 \\ 1 \\ 0 \\ -1 \end{pmatrix}. \tag{2}$$

Eliminating x is equivalent to multiplying A and b on the left with non negative matrix R where $R = \begin{pmatrix} 1 & 1/2 & 0 & 0 \\ 0 & 1/2 & 0 & 1/4 \\ 1 & 0 & 1/3 & 0 \\ 0 & 0 & 3 & 9/4 \end{pmatrix}$. We now get the following:

$$\begin{aligned} 5/2\ y - 1/2\ z &\leq 5/2 \\ -5/4\ y + 1/4\ z &\leq 1/4 \\ 10/3\ y - 2/3\ z &\leq 2 \\ -15/4\ y + 3/4\ z &\leq -9/4 \end{aligned}$$

Note that the second inequality generated above is redundant because it is implied by the fourth inequality generated. The third and fourth inequalities generated above imply that $z = 5y - 3$. The third inequality above is generated by eliminating x from the original first and third inequalities while the fourth inequality above is generated from the original first and fourth inequalities. It is easily verified that the original first, third and fourth inequalities imply that $z = 5y - 3$.

Substituting this in the original set of inequalities in S gives:

$$\begin{aligned} x - y &\leq -1 \\ -2x + 2y &\leq 4 \\ -3x + 3y &\leq 3 \\ 4x - 4y &\leq -4 \end{aligned}$$

Note that the fourth inequality generated is redundant as it is equivalent to the first inequality generated. The second inequality generated is dominated by the third inequality generated.

The first and third inequalities generated above together define an implicit equation $x - y = -1$. If we substitute $z = 5y - 3$ in the original four inequalities, the first and the third inequalities imply that $x - y = -1$. If $y = 1$, then $z = 2$ which implies that $\{x \leq 0,\ -2x \leq 2,\ -3x \leq 0,\ 4x \leq 0\} \implies x = 0$.

Example 3 Consider the system of inequalities

$$\{x + 4y - z \leq 2,\ -10y + 2z \leq -8, -3x - 2y + z \leq 0, 4x + y - z \leq -1\}$$

On eliminating x, when the first and third inequalities are considered together, we get $10y - 2z \leq 6$, but the second inequality constraint states $10y - 2z \geq 8$ which clearly implies that there exist no feasible values for y and z. Thus this system is not solvable.

2.2 Projection: the geometry of elimination

We saw earlier that Fourier elimination of a variable in a linear inequality system actually constructs the projection or shadow of the convex polyhedron in the space that is diminished in dimension by one. Not surprisingly, the projection of a convex polyhedron is another convex polyhedron as described by the system of linear inequalities produced by the Fourier construction.

It is natural to wonder if elimination of a block of variables can be executed simultaneously- rather than one variable at a time. Indeed this is possible and in fact leads to a technique that is a much improved elimination method.

First let us identify the set of variables to be eliminated. Let the input system be of the form

$$P = \{(x, u) \in \mathbb{R}^{n_1+n_2} | Ax + Bu \leq b\}$$

where u is the set to be eliminated. The projection of P onto x or equivalently the effect of eliminating the u variables is

$$P_x = \{x \in \mathbb{R}^{n_1} | \exists u \in \mathbb{R}^{n_2} \text{ such that } Ax + Bu \leq b\}.$$

Now W, the projection cone of P, is given by

$$W = \{w \in \mathbb{R}^m | B^t w = 0, w \geq 0\}.$$

A simple application of Farkas Lemma yields a description of P_x in terms of W.

Projection lemma Let G be any set of generators (e.g. the set of extreme rays) of the cone W. Then $P_x = \{x \in \mathbb{R}^{n_1} |\ (gA)x \leq gb\ \ \forall g \in G\}$.

The lemma, sometimes attributed to Cerkinov, reduces the computation of P_x to enumerating the extreme rays of the cone W or equivalently the extreme points of the polytope $W \cap \{w \in \mathbb{R}^m |\ \sum_{i=1}^m w_i = 1\}$. We will see in a later section that Fourier elimination can be used to solve this problem.

2.3 Polynomial solvability

Theoretical computer science has provided us with frameworks for the classification of the inherent complexity of a problem class such as linear programming problems along with precise ways of analysing the time and space requirements of specific algorithms. The notion of "polynomial solvability" of linear programming has become synonymous with "tractability". The following definitions are useful in studying the post 1970s literature on linear programming:

- *Encoding* The length of a problem instance is defined to be the length of the description of the problem instance to a Turing Machine with the rational numerical coefficients encoded in binary fixed base notation.
- *Polynomial time solvability* A problem class is said to be polynomial time solvable or in PTIME if there is an algorithm for this problem that runs in time that is bounded above by a polynomial function of the length of encoding of the problem instance. Note that the space used by the algorithm would also need to be bounded by a polynomial in the length of the encoding.
- *Arithmetic model* A more natural and realistic model of computation is to consider elementary arithmetic operations such as addition, subtraction, multiplication, division and

comparison as atomic operations and we use the total count of the number of such arithmetic operations as the time complexity of an algorithm.

- *Strongly polynomial time* We say that an algorithm is strongly polynomial time if it is a polynomial space algorithm and the number of arithmetic operations it performs is bounded by a polynomial function of the number of input numbers. So this definition mixes the Arithmetic and Turing models of computation.

Much of the remaining sections of this paper will focus on polynomial time solvability of linear programming. Let us first examine Fourier's elimination method from this perspective. It is important to note that Fourier's algorithm is independent of the order in which the variables are eliminated. The number of steps necessary to eliminate all variables, however, may vary depending on this order.

Fourier elimination solves our LP problem instance, however it is not polynomial. In the worst case, the Fourier elimination method can be quite inefficient. To perform an elimination over n inequalities leads to at most $n^2/4$ inequalities in the output. Running k successive steps leads to $(n^2/4)^k$. This explosion in the number of inequalities can be observed on a wide variety of problems in practice and is a manifestation of the problem of intermediate swell that plagues many algorithms in symbolic computation.

Let us re-examine the stopping conditions of Fourier elimination in terms of $\tilde{b}$.

(i) ($\tilde{b} \ngeq 0$): the system $Ax \leq b$ is unsolvable.
(ii) (Undefined $\tilde{b}$): When all the inequalities are deleted in the first step itself while trying to eliminate some variable, $\tilde{b}$ is undefined and the system $Ax \leq b$ is strongly solvable.
(iii) ($\min \tilde{b_i} > 0$): the system $Ax \leq b$ is solvable and the polyhedron is full-dimensional.
(iv) ($\min \tilde{b_i} = 0$): the system $Ax \leq b$ is solvable and contains implicit equations.

We will now take a closer look at these stopping conditions and see that they each provide useful insights on the structure of the underlying polyhedron $\{x \in \mathbb{R}^n : Ax \leq b\}$

2.4 Unsolvability and Farkas lemma

When presented with an unsolvable system of linear inequalities, Farkas lemma is a simple consequence of analysing the stopping condition of Fourier elimination. The lemma states the remarkable property that the unsolvability of a system of linear inequalities is directly related to the solvability of a 'dual' system.

A form of Farkas lemma that will be used in a later section states that:

Lemma 2.1 *Exactly one of the alternatives*

$$1.\ \exists x \in \mathbb{R}^n_+ : Ax \leq b \qquad 2.\ \exists y \in \mathbb{R}^m_+ : y^t A \geq 0,\ y^t b < 0$$

is true for any given real matrices A and b.

Proof 1 is equivalent to the system of inequalities $Ax \leq b$ and $-Ix \leq 0$ where I is a $n * n$ identity matrix. Let

$$A^* = \begin{pmatrix} A \\ I \end{pmatrix} \tag{3}$$

and

$$b^* = \begin{pmatrix} b \\ 0 \end{pmatrix}. \tag{4}$$

Let us analyse the case when Fourier elimination provides a proof of the unsolvability of a given linear inequality system $A^*x \leq b^*$. The method clearly converts the given system into

$RA^*x \leq Rb^*$ where R is a non-negative matrix, $RA^* = 0$ and Rb^* has at least one negative component. Partition $R = [R_1\ R_2]$ corresponding to the partitioning of

$$A^* = \begin{pmatrix} A \\ I \end{pmatrix}. \tag{5}$$

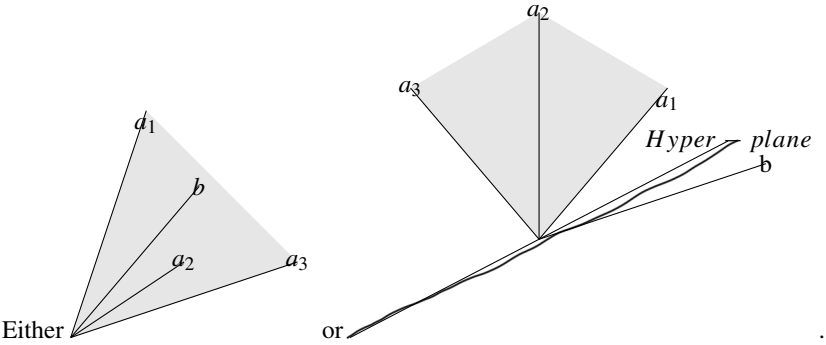
So $RA^* = 0 \implies R_1A - R_2I = 0$. Therefore there is some row of the non negative matrix R, say $r \in \mathbb{R}_+^{m+n}$, such that $rA^* = 0$ and $rb^* < 0$. Partitioning $r = [r_1\ r_2]$ corresponding to the partitioning of A^* as above leads to $rA^* = 0 \implies r_1A - r_2I = 0$. Since the elements of the row vector r_2 are non-negative, it follows that the row vector $r_1A \geq 0$. Similarly, partitioning b^* as above, we have that $rb^* < 0 \implies r_1b < 0$. Thus $1 \implies 2$.

Next, we show that both 1 and 2 cannot be simultaneously true for fixed A and b. Assume the contrary. We have $x^* \geq 0,\ Ax^* \leq b$ and $y^* \geq 0,\ y^{*t}A \geq 0,\ y^{*t}b < 0$. Now multiplying $Ax^* \leq b$ on the left by $y^{*t} > 0$ we have that $y^{*t}Ax^* \leq y^{*t}b < 0$. But since $y^{*t}A \geq 0$ and $x^* \geq 0$ we have that the left hand side of the inequality $y^{*t}Ax^* \leq y^{*t}b$ is ≥ 0 while the right hand side is less than 0 which is a contradiction. This completes the proof of the theorem. □

Geometric interpretation. Let a_i be the ith column of A. Farkas lemma states that exactly one of the following statements is true:

1. $\exists x_1, x_2, \ldots, x_n \in \mathbb{R}_+$: $x_i \geq 0\ \forall i$ and $a_1x_1 + \ldots + a_nx_n \leq b$.
2. $\exists y \in \mathbb{R}_+^m : a_i^t y \geq 0\ \forall i$ and $b^t y < 0$.

The convex cone of the set $\{a_1, a_2, \ldots, a_n\}$ is represented by $a_1x_1 + \ldots + a_nx_n$ where each $x_i \geq 0$. Therefore, the first statement says that the vector b lies in this cone. The second statement says that there exists a vector y such that the angle it makes with a_i is at most 90° while the angle between y and b is greater than 90°. The hyperplane which is normal to this vector separates the vectors of the convex cone and the vector b.



2.5 Strong solvability

As defined earlier, a solvable system of inequalities is said to be strongly solvable if any modification of the right hand side constants of the inequalities results in a solvable set. This is a valuable property to detect since in one fell swoop we would have the answer to an entire family of solvability problems. When this occurs, it is only due to the left hand side of the inequalities, not the right hand side constants. Changing these constants does not modify the outcome.

In the Fourier elimination method, the final $\tilde{b}$ may not be defined if all the inequalities are deleted by the monotone sign condition while trying to eliminate some variable. In such a

situation we declare the system $Ax \leq b$ strongly solvable. This is a valid conclusion since, regardless of what $b \in \mathbb{R}^m$ is chosen, the last variable to be eliminated, say x_j has only upper (lower) bounds depending on whether it appears with a positive (negative) coefficient in all the remaining inequalities. Thus for any set of values assigned to the other un-eliminated variables, we can always choose a small (large) enough value for x_j that solves the system.

In geometric terms, if the right hand side constants are modified, it corresponds to a translation of the hyperplane associated to the inequality. So the strong solvability property means that any translation of those hyperplanes results in a new non-empty polyhedron. Another geometric characterisation of strong solvability states that a system of linear inequalities $S = \{Ax \leq b\}$ is strongly solvable if and only if the polyhedron $P = \{x : Ax \leq b\}$ contains spheres of unbounded radii.

In the first example, after the first round of iteration (i.e. after eliminating x_1), we had obtained the following inequalities: $x_2 \geq 3$ and $x_2 \geq \frac{15}{7}$. As x_2 appears with a positive coefficient in all the inequalities, the Fourier's algorithm returns the empty set. Hence the system is actually strongly solvable.

2.6 Implicit equations

An important problem in symbolic computation with linear constraints is the identification of implicit equations in a system of linear inequalities. In geometric terms, we need the affine hull of the convex polyhedron represented by the linear inequalities. A linear combination is affine when the multipliers add to 1 and the affine hull of a set of points is the collection of all affine combinations of the set.

Jean-Louis Laseez and Michael Maher made the interesting observation that the detection of implicit equations by Fourier elimination is easily accomplished. Run the Fourier method until all variables have been eliminated. The implicit equalities are exactly those original constraints used in producing $\tilde{b_i} = 0$. When S is solvable and the Fourier's algorithm produces at least one tautology of the form $0 \leq 0$, implies that S contains at least two inequalities which imply at least one equality. If an initial set of constraints does not contain implicit equalities, none of the sets generated by the elimination of variables will contain implicit equalities. In a geometric context, the projections of a full dimensional polyhedral set are also full dimensional. This helps avoid unnecessary computation in a problem of redundancy removal.

This is best illustrated through an example.

Example Consider the system S of linear inequalities

$$\{2x_1 - x_2 + x_3 \leq 2, x_1 - 3x_2 \leq -2, -x_1 + x_2 \leq 0, x_2 \leq 1\}$$

To eliminate x_1, the second and third inequalities generate the fifth inequality $2 \leq 2x_2$. While eliminating x_2, the fourth and the fifth inequalities generate $0 \leq 0$ which results in $\min_i\{\tilde{b_i}\} = 0$. Thus the second, third and fourth inequalities actually define implicit equations, $x_1 = 1$ and $x_2 = 1$. Continuing further, we get $\{x_3 \leq 1\}$. The polyhedron P is thus defined by $\{(x_1, x_2, x_3) : x_1 = x_2 = 1,\ x_3 \leq 1.\}$ The affine hull of $P = \{(x_1, x_2, x_3) : x_1 = x_2 = 1\}$ (Fig. 1) (Original figure taken from "The French Connection" by Vijay Chandru and M R Rao, Resonance, Journal of Science Education, Vol 3, Issue 10, October 1998, pp 26–40).

2.7 An application of the Fourier method

The backbone of the famous Hochbaum and Naor algorithm Hochbaum and Naor (1994) is the Fourier–Motzkin elimination method. The algorithm solves feasibility in linear programs

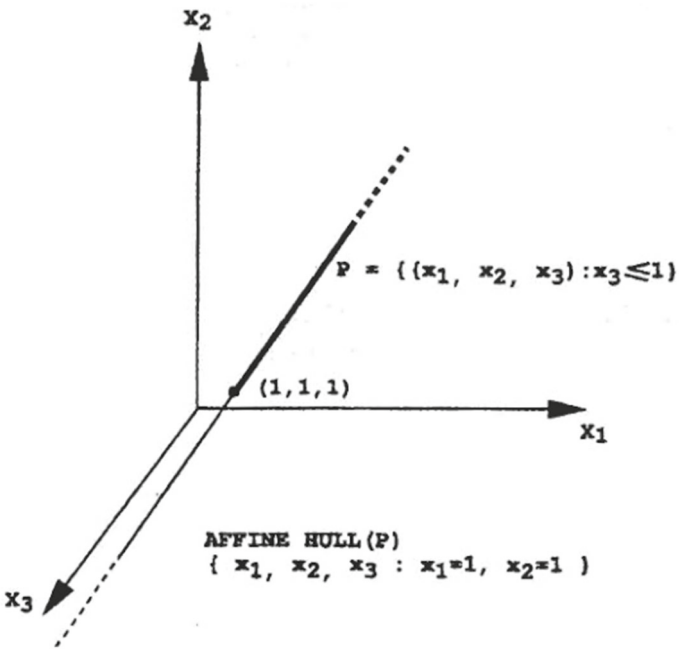


Fig. 1 Graph represented by S

with up to two variables per inequality derived directly from Fourier's method. The running time is $\mathcal{O}(mn^2 \log m)$ where m and n are the number of variables and inequalities respectively. In general, the algorithm does not run in polynomial time because while eliminating variables, exponential number of inequalities could be generated. However, it is very efficient for linear programs where each inequality may contain at most two variables.

An equivalent representation of the linear program is by the graph $G = (V, E)$, where $V = \{x_0, x_1, \ldots, x_n\}$ and an edge between vertex x_i and x_j represents the set of inequalities involving them. The vertex x_0 is needed to represent those inequalities that contain just one variable.

The main feature that allows for the efficient implementation of the Fourier–Motzkin algorithm is the following: the set of inequalities that correspond to an edge between x_i and x_j is represented in the (x_i, x_j) plane as two envelopes, an upper and a lower. The feasible region of x_i and x_j is in between the two envelopes and each envelope is a piecewise linear function that can be represented by its breakpoints.

At step i of the Fourier elimination, the algorithm is as follows:

Let G_i denote the graph corresponding to the linear program E_i (set of inequalities at step i which will only contain variables $x_i, \ldots, x_n$) and $x_i^{\min}$ ($x_i^{\max}$) denote the minimum (maximum) feasible values of x_i. The procedure given by Nimrod Megiddo (1983) gives an efficient way to find $x_i^{\min}$ and $x_i^{\max}$.

1. Let the neighbours of x_i in the graph G_i be $x_{i_1}, \ldots, x_{i_d}$ and $B_j (1 \leq j \leq d)$ denote the set of breakpoints of the edge (x_i, x_{i_j}) projected on the x_i coordinate.
2. Merge the sorted sequences B_i into a sorted sequence B where the sorted sequence is $b_1, \ldots, b_k$.
3. A binary search is performed on B for either

(a) a breakpoint $b_l \in B$ such that $x_i^{\min} \leq b_l \leq x_i^{\max}$ or
(b) an interval $[b_l, b_{l+1}]$ $(1 \leq l \leq k)$ such that $b_l < x_i^{\min}$ and $x_i^{\max} < b_{l+1}$.

4. In step 3a, $x_i = b_l$ and contracted with vertex x_0 in graph G_i. In step 3b, the number of inequalities on each edge adjacent to x_i is reduced to at most two. Now, the Fourier method is applied to variable x_i.

The number of breakpoints on an edge is at most $\mathcal{O}(m)$ and the complexity of eliminating a variable in the algorithm is $\mathcal{O}(mn \log m)$. Hence, the complexity of the entire algorithm is $\mathcal{O}(mn^2 \log m)$.

The same paper by Hochbaum and Naor (1994) also presents an application of the Fourier algorithm to identify fat polytopes i.e. whether it contains a sphere circumscribing a unit hypercube. Since a unit hypercube must contain at least one integer lattice point, an integer feasible point is found by rounding the coordinates of the center of the sphere to the nearest integer, however there is a possibility that there may exist a feasible integer point but the polytope does not contain a large enough sphere. To avoid this problem, all constraints are shifted by a distance of r. In order to obtain a sphere large enough to contain a unit hypercube, set $r = \sqrt{n}/2$. Consequently, an integer feasible point can be found out in $\mathcal{O}(mn^2 \log m)$ time.

3 Linear programming: a brief description of the simplex method

A pointed polyhedron is one that does not contain a line that is infinite in both directions. The feasible region of a canonical linear program $\mathcal{K} = \{x \in \mathbb{R}^n : Ax = b,\ x \geq 0\}$ is a *pointed* convex polyhedron. Hence, if $\mathcal{K} \neq \emptyset$ then $\mathcal{K}$ has at least one extreme point.

Theorem of linear programming states that if $\min \{cx : Ax = b,\ x \geq 0\}$ has an optimal solution, then it has an optimal extreme point solution. A *face* of the polyhedron $\mathcal{K}$ is a lower dimensional polyhedron contained in K defined by forcing a subset of x-variables to 0. The *adjacency graph* of the polyhedron is a graph whose vertices are the extreme points of the polyhedron and whose edges are the 1-dimensional faces of the polyhedron. The search for an optimal solution to is a search over the extreme points or vertices of the graph. A simple local improvement search strategy of moving from extreme point to an adjacent extreme point until we get to a local optimum is nothing but the simplex method of linear programming (Dantzig 1951, 1963). The local optimum also turns out to be a global optimum, because of the convexity of the polyhedron $\mathcal{K}$ and the linearity of the objective function cx.

Procedure: **Simplex** $(\mathcal{K}, c)$

0. **Initialize:**

- x_0 := an extreme point of $\mathcal{K}$
- $k := 0$

1. **Iterative Step:**

do
If for all edge directions $\mathcal{D}_k$ at x_k, the objective function is nondecreasing, i.e.,

$$cd \geq 0 \quad \forall\, d \in \mathcal{D}_k$$

then exit and return optimal x_k.
Else pick some d_k in $\mathcal{D}_k$ such that $cd_k < 0$.
If $d_k \geq 0$ **then** declare the linear program unbounded in objective value and exit.
Else $x_{k+1} := x_k + \theta_k * d_k$, where

$$\theta_k = \max \{\theta : x_k + \theta * d_k \geq 0\}$$

$k := k + 1$
od

2. **End**

We assumed that an extreme point x_0 of the polyhedron $\mathcal{K}$ is available. A general technique for generating a starting extreme point is to solve a Phase I model

$$\min \{v_0 : Ax + bv_0 = b,\ x \geq 0,\ v_0 \geq 0\}$$

for which $(x, v_0)^T = (0, \ldots, 0, 1)$ is a self-evident extreme point, and $v_0 = 0$ at an optimal solution of this model is a necessary and sufficient condition for the solvability of $Ax = b,\ x \geq 0$.

The source of non-determinism in the algorithm arises from geometric degeneracy. When there are multiple feasible bases corresponding to an extreme point, the simplex method has to pivot from basis to adjacent basis by picking an entering basic variable (a pseudo edge direction) and by dropping one of the old ones. A wrong choice of the leaving variables may lead to cycling in the sequence of feasible bases generated at this extreme point. The techniques of Dantzig et al. (1955) and Charnes (1952) were first used to prove finite termination of the simplex method. A clever method proposed by Bland (cf. Schrijver 1986) preorders the rows and columns of the matrix A. In case of non-determinism in either entering or leaving variable choices, Bland's Rule just picks the lowest index candidate. All cycles are avoided by this rule also. Bland's rule has not been found to be efficient. Moreover, in practice cycling does not seem to happen but stalling does. A technique due to Wolfe (1963) seems to avoid stalling in practice.

Due to space limitations, one aspect of linear programming that we have not elaborated on is the robustness issue or the effect of small perturbations of the data on the solution. There are both practical issues of numerical stability in factorisations as well as algorithmic issues of convergence that arise because of this. Most advanced textbooks in linear programming will provide discussion of these issues.

3.1 Simplex method is worst-case exponential

There is no known variant of the simplex method with a worst-case polynomial guarantee on the number of iterations. This despite the fact that both empirical (Bixby 1994; Dantzig 1963) and probabilistic (Borgwardt 1987; Haimovich 1983) analyses indicate that the number of iterations of the simplex method is just slightly more than linear in the dimension of the primal polyhedron. So the expected behaviour of the Simplex Method is strongly polynomial.

However, examples of exponential performance have been created. Klee and Minty (1972) exploited the sensitivity of the original simplex method of Dantzig, to projective scaling of the data, and constructed exponential examples for it. These examples were simple projective distortions of the hypercube to embed long isotonic (improving objective value) paths in

the graph. Scaling is used in the Klee–Minty construction, to trick the choice of entering variable (based on most negative reduced cost) in the simplex method and thus keep it on an exponential path. Later, several variants of the entering variable choice (best improvement, steepest descent, etc.) were all shown to be susceptible to similar constructions of exponential examples (cf. Schrijver 1986).

The ellipsoid method of Shor (1970), specialized for approximate convex optimization by Nemirovsky and Yudin in 1972, was devised to overcome poor scaling in convex programming problems and therefore turned out to be the natural choice of an algorithm to first establish polynomial-time solvability of linear programming, as demonstrated by Khachiyan in 1979. Later Karmarkar (1984) took care of both projection and scaling simultaneously and arrived at a superior algorithm.

3.2 The linear programming duality theorem

Building on polarity in cones and polyhedra, duality in linear programming is a fundamental concept which is related to both the complexity of linear programming and to the design of algorithms for solvability and optimisation. If we take the *primal* linear program to be

$$\text{(P)} \qquad \max_{x \in \mathbb{R}^n} \{cx : Ax \leq b, x \geq 0\}$$

there is an associated *dual* linear program

$$\text{(D)} \qquad \min_{y \in \mathbb{R}^m} \{b^t y : A^t y \geq c^t, y \geq 0\}$$

and the two problems are related through several properties that we now recount. Note that by symmetry the dual of (D) is (P).

Proposition 3.1 *(Weak Duality) For any $\hat{x}$ and $\hat{y}$ feasible in (P) and (D) we have $c\hat{x} \leq b^t \hat{y}$.*

Proof $c\hat{x} \leq \hat{y}^t A\hat{x} \leq \hat{y}^t b$ where the first inequality holds from (D)-feasibility of $\hat{y}$ and non-negativity of $\hat{x}$, and the second inequality follows from (P)-feasibility of $\hat{x}$ and non-negativity of $\hat{y}$. □

Note that if (P) is unbounded then (D) is infeasible and similarly if (D) is unbounded then (P) is infeasible. Consequently both (P) and (D) cannot be unbounded. This however leaves the possibility that both (P) and (D) are infeasible. An example of this possibility would be: maximise $2x_1 - x_2$ subject to $x_1 - x_2 \leq 1,\ -x_1 + x_2 \leq -2, x_1, x_2 \geq 0$. See Vanderbei (2015).

Proposition 3.2 *If (P) has a finite optimal solution then (D) has a feasible solution.*

Proof Suppose (P) has an optimal solution x^* with $cx^* = z^*$. Let $\bar{z} > z^*$. Then $max\{cx : Ax \leq b,\ cx \geq \bar{z},\ x \geq 0\}$ has no feasible solution. Now consider the two inequality systems

$$\text{(I)} \quad \{Ax \leq b,\ -cx \leq -\bar{z},\ x \geq 0\}$$
$$\text{(II)} \quad \{y^t A - \lambda c \geq 0,\ y^t b - \lambda \bar{z} < 0,\ y \geq 0, \lambda \geq 0\}$$

By Farkas lemma, since (I) is not feasible, (II) must be feasible. Let $\bar{y}^t$ and $\bar{\lambda}$ be a feasible solution to (II). Then

$$\bar{y}^t A - \bar{\lambda} c \geq 0,\ \bar{y}^t b - \bar{\lambda}\bar{z} < 0,\ \bar{y} \geq 0,\ \bar{\lambda} \geq 0$$

There are two cases to consider:

1. $\bar{\lambda} = 0$: Now $\bar{y}^t A \geq 0$ and $\bar{y}^t b < 0$. Now consider the pair of dual programs

$$\begin{aligned} &\text{(P1)} \quad max\{0x : Ax \leq b,\ x \geq 0\} \\ &\text{(D1)} \quad min\{y^t b : A^t y \geq 0,\ y \geq 0\} \end{aligned}$$

   Since $\bar{y}^t$ is feasible to (D1) and $\bar{y}^t b < 0$, it follows that (D1) is unbounded. Consequently (P1) must be infeasible which is a contradiction to the assumption that (P) has a finite optimum.
2. $\bar{\lambda} > 0$: Since (II) is a homogeneous system, we can take $\bar{\lambda} = 1$. Now $\bar{y}^t A \geq c,\ \bar{y}^t b < \bar{z}, \bar{y} \geq 0, \bar{\lambda} = 1$. Clearly, $\bar{y}^t$ is feasible to (D) and the proposition follows.

□

The weak duality condition gives us a technique for obtaining lower bounds for minimisation problems and upper bounds for maximisation problems. Hence,

Corollary 3.1 *The linear program (P) has a finite optimal solution if and only if its dual (D) does.*

If the linear programs have finite optima, the inequality of the weak duality relation can be strengthened to an equality.

Theorem 3.1 (Strong duality) *x^* and y^* are a pair of optimal solutions for (P) and (D) respectively, if and only if x^* and y^* are feasible in (P) and (D) and $cx^* = b^t y^*$.*

Proof Let x^* and y^* be a pair of feasible solutions to (P) and (D) respectively that satisfy $cx^* = b^t y^*$. It follows from the weak duality proposition that x^* and y^* are a pair of optimal solutions for (P) and (D) respectively.

To prove the 'only if' direction of the theorem we assume that (P) and (D) are optimised by x^* and y^* respectively. Consider the two inequality systems.

$$\begin{aligned} &\text{(I)} \quad \{Ax \leq b, -A^t y \leq -c^t \ \ b^t y - cx \leq 0,\ -x \leq 0, -y \leq 0\}. \\ &\text{(II)} \quad \{A^t \alpha - c^t \lambda \geq 0, -A\beta + b\lambda \geq 0, b^t \alpha - c\beta < 0, \alpha \geq 0, \beta \geq 0, \lambda \geq 0\} \end{aligned}$$

From Farkas Lemma, we know that (I) or (II) must be solvable but not both. We demonstrate below that (I) must be solvable since (II) is unsolvable. But then it follows that (x^*, y^*) must solve (I) for if it does not, then it follows that $b^t y^* > cx^*$. Suppose now $(\bar{x}, \bar{y})$ solves (I), then $b^t \bar{y} \leq c\bar{x}$ and this implies that $c\bar{x}$ would get above cx^* or $b^t \bar{y}$ would get below $b^t y^*$. Since $\bar{x}$ and $\bar{y}$ are (P) and (D) feasible, this would contradict the optimality of x^* and y^*.

It therefore remains to show that the inequality system (II) is unsolvable. Suppose (α, β, λ) is a solution to (II). Let us consider two cases below:

1. $(\lambda = 0)$: In this case $A^t \alpha \geq 0, -A\beta \geq 0, b^t \alpha - c\beta < 0, \alpha \geq 0, \beta \geq 0$. If $b^t \alpha \geq 0$ then $c\beta > 0$ and we have

$$\begin{aligned} &(P_1) \quad \max\ \{c\beta : A\beta \leq 0, \beta \geq 0\} \\ &(D_1) \quad \min\ \{0\alpha : A^t \alpha \geq c^t,\ \alpha \geq 0\} \end{aligned}$$

   where (P_1) is unbounded. Then (D_1) should be infeasible which contradicts the assumption that (D) is feasible. Conversely if $b^t \alpha < 0$ then we have

$$\begin{aligned} &(P_2) \quad \max\ \{0\beta : A\beta \leq b, \beta \geq 0\} \\ &(D_2) \quad \min\ \{b^t \alpha : A^t \alpha \geq 0, \alpha \geq 0\} \end{aligned}$$

   where (D_2) is unbounded. Then its corresponding (P_2) should be infeasible which is a contradiction to the assumption that (P) is feasible. Therefore this case is ruled out.

2. ($\lambda > 0$): Since system (II) is homogeneous, we may as well take $\lambda = 1$. But then (β, α) are feasible in (P), (D) with $b^t\alpha - c\beta < 0$ which is impossible since it violates weak duality. Hence this case is also ruled out. □

Remark From the above proof it is evident that we can simultaneously optimise (P) and (D) by solving the system of inequalities (I). Therefore the solvability of linear inequalities subsumes linear optimisation.

The strong duality condition above gives us a good stopping criterion for optimisation algorithms. It would be useful to have constructions for moving from dual to primal solutions and vice-versa. The necessary and sufficient conditions for optimality which follow from strong duality as given below, provide just that.

Complementary slackness: x^* and y^* are a pair of optimal solutions for (P) and (D) respectively, if and only if x^* and y^* are feasible in (P) and (D) and $(Ax^* - b)^t y^* = (A^t y^* - c)^t x^* = 0$.

Note that the properties above have been stated for linear programs in a particular form. The reader should be able to check, that if for example the primal is of the form

$$(\mathrm{P}') \qquad \min_{x \in \mathbb{R}^n} \{cx : Ax = b, x \geq 0\}$$

then the corresponding dual will have the form

$$(\mathrm{D}') \qquad \max_{y \in \mathbb{R}^m} \{b^t y : A^t y \leq c^t\}.$$

The tricks needed for seeing this are that any equation can be written as two inequalities, an unrestricted variable can be substituted by the difference of two non-negatively constrained variables and an inequality can be treated as an equality by adding a non-negatively constrained variable to the lesser side. Using these tricks, the reader could also check that the dual construction in linear programming is involutory (i.e. the dual of the dual is the primal).

3.3 Implicit and parametric representations

A system of linear inequalities of the form $Ax \leq b$ represents a convex polyhedron K, implicitly. It is implicit in that we are given the bounding half-spaces and the representation does not directly provide a scheme for generating points in K. An explicit or parametric representation of K would require the lists of extreme points $\{p^1, p^2, \ldots, p^K\}$ and extreme rays $\{r^1, r^2, \ldots, r^L\}$ of K. And then the convex multipliers $\{\alpha_1, \alpha_2, \ldots \alpha_K\}$ and the positive cone multipliers $\{\mu_1, \mu_2, \ldots, \mu_L\}$ are the parameters that give us the representation:

$$K = \left\{ x \in \mathbb{R}^n : x = \sum_{i=1}^{K} \alpha_i p^i + \sum_{j=1}^{L} \mu_j r^j, \sum_{i=1}^{K} \alpha_i = 1, \alpha_i \geq 0 \ \forall i, \ \mu_j \geq 0 \ \forall j \right\}$$

An organic role of Fourier elimination is in obtaining an implicit representation of a convex polyhedron from a parametric representation. The parametric representation above is a system of linear equations and inequalities in x, α and μ. If we eliminate the α and μ variables from this system, we would obtain an implicit representation.

The converse problem of generating a parametric representation from a given implicit representation of a polyhedron can also be attacked with Fourier elimination. In an intriguing paper, Paul Williams (1986) shows that a dual interpretation of Fourier elimination yields a

scheme for enumerating the extreme rays and extreme points of a polyhedron defined by a system of linear constraints $\{Ax \leq b, x \geq 0\}$.

Picking an arbitrary $c^t \in \mathbb{R}^n$, we start with the dual pair of linear programs

$$\begin{aligned} &(\mathrm{P}) \quad \max \{cx : Ax \leq b, x \geq 0\} \\ &(\mathrm{D}) \quad \min \{b^t y : A^t y \geq c^t, y \geq 0\} \end{aligned}$$

Now introduce a new variable z in (D) to get

$$(\mathrm{D}') \quad \min \{z : z - b^t y \geq 0,\ A^t y \geq c^t, y \geq 0\}$$

The linear program dual to (D′) is given by

$$(P') \quad \max \{cx : Ax - b\alpha + \beta = 0, \alpha = 1, x \geq 0, \alpha \geq 0, \beta \geq 0\}$$

Using Fourier elimination on the constraints of (D′) we eliminate all y variables and are left with bounds (lower and upper) on z. Fourier elimination on (D′) takes positive combinations of the constraints to eliminate y variables but at the cost, in general, of many new constraints. In (P′) this corresponds to column operations to eliminate rows (constraints) at the cost of generating many new columns (variables).

At the completion of the elimination of y variables in (D′), we have all constraints of (P′) eliminated except for the transformed equation representing the original normalising constraint $\alpha = 1$ and non negativity restrictions on the transformed variables. The extreme points and rays of the polyhedron defined by a single equation on non negative variables can be simply read off. If we revert the column operations to interpret these extreme points and rays in terms of the original x variables we will obtain the extreme points and rays of the polyhedron defined by the constraints of (P). Unfortunately, as seen in the example below, we may also obtain some non-extreme solutions which need to be recognised and discarded.

To get the extreme points of the polyhedron

$$K = \{(x_1, x_2) : -x_1 + 2x_2 \leq 3, 2x_1 - 7x_2 \leq -15, x_1 \geq 0, x_2 \geq 0\}$$

we define the pair of linear programs (P′) and (D′) as above. Using Fourier's method to eliminate the y variables in (D′) we wend up with the transformed version of (P′) whose constraints have the form

$$\{2v_1 + v_2 + 0v_3 + 0v_4 = 1, v_j \geq 0, j = 1, 2, 3, 4\}.$$

The extreme points are read off as $(1/2, 0, 0, 0)$ and $(0, 1, 0, 0)$ and the two extreme rays as $(0, 0, 1, 0)$ and $(0, 0, 0, 1)$. Inverting the column operations we obtain $(3, 3)$ and $(10, 5)$ as the candidate extreme points of K, and $(7, 2)$ and $(2, 1)$ as the candidate extreme rays of K. $(10, 5)$ is not a corner point for K.

As mentioned earlier, working with the extreme points of the feasible region of a linear program is crucial since if an optimal solution exists then it does so at the extreme point. Searching through all the extreme points for the most optimal one is highly laborious. The simplex method is one way to execute a partial search to zero in on an optimal extreme point.

4 Khachiyan's method: the ellipsoid algorithm for linear programming

The Ellipsoid Algorithm of Shor (1970), specialized for approximate convex optimization by Nemirovsky and Yudin in 1972, gained prominence in the late 1970s when Haĉijan

(pronounced Khachiyan) Haĉijan (1979) showed that this convex programming method specialises to a polynomial-time algorithm for linear programming problems. This theoretical breakthrough naturally led to intense study of this method and its properties. The direct theoretical consequences for combinatorial optimisation problems was independently documented by Padberg and Rao (1981), Karp and Papadimitriou (1982), and Grötschel et al. (1982). The ability of this method to implicitly handle linear programs with an exponential list of constraints and maintain polynomial-time convergence is a characteristic that is the key to its applications in combinatorial optimisation.

Let us consider the problem of testing if a polyhedron $\mathcal{Q} \in \mathbb{R}^d$, defined by linear inequalities, is nonempty. For technical reasons let us assume that $\mathcal{Q}$ is rational, i.e., all extreme points and rays of $\mathcal{Q}$ are rational vectors or equivalently that all inequalities in some description of $\mathcal{Q}$ involve only rational coefficients. The ellipsoid method does not require the linear inequalities describing $\mathcal{Q}$ to be explicitly specified. It suffices to have an oracle representation of $\mathcal{Q}$. Several different types of oracles can be used in conjunction with the ellipsoid method (Grötschel et al. 1988; Karp and Papadimitriou 1982; Padberg and Rao 1981. We will use the *strong separation oracle* described below.

Oracle: **Strong Separation** $(\mathcal{Q}, y)$
Given a vector $y \in \mathbb{R}^d$, decide whether $y \in \mathcal{Q}$, and if not find a hyperplane that separates y from $\mathcal{Q}$; more precisely, find a vector $c \in \mathbb{R}^d$ such that $c^T y < \min\{c^T x : x \in \mathcal{Q}\}$.

The ellipsoid algorithm initially chooses an ellipsoid large enough to contain a part of the polyhedron $\mathcal{Q}$ if it is nonempty. This is easily accomplished because we know that if $\mathcal{Q}$ is nonempty then it has a rational solution whose (binary encoding) length is bounded by a polynomial function of the length of the largest coefficient in the linear program and the dimension of the space.

The center of the ellipsoid is a feasible point if the separation oracle tells us so. In this case, the algorithm terminates with the coordinates of the center as a solution. Otherwise, the separation oracle outputs an inequality that separates the center point of the ellipsoid from the polyhedron $\mathcal{Q}$. We translate the hyperplane defined by this inequality to the center point. The hyperplane slices the ellipsoid into two halves, one of which can be discarded. The algorithm now creates a new ellipsoid that is the minimum volume ellipsoid containing the remaining half of the old one. The algorithm questions if the new center is feasible and so on. The key is that the new ellipsoid has substantially smaller volume than the previous one. When the volume of the current ellipsoid shrinks to a sufficiently small value, we are able to conclude that $\mathcal{Q}$ is empty. This fact is used to show the polynomial time convergence of the algorithm.

Ellipsoids in $\mathbb{R}^d$ are denoted as $E(A, y)$ where A is a $d \times d$ positive definite matrix and $y \in \mathbb{R}^d$ is the center of the ellipsoid $E(A, y)$.

$$E(A, y) = \left\{ x \in \mathbb{R}^d \;\middle|\; (x - y)^T A^{-1} (x - y) \leq 1 \right\}.$$

The ellipsoid algorithm is described on the iterated values, A_k and x^k, which specify the underlying ellipsoids $E_k(A_k, x^k)$.

Procedure: **Ellipsoid** ($\mathcal{Q}$)

0. **Initialize:**

- $N := N(\mathcal{Q})$ (comment: iteration bound)
- $R := R(\mathcal{Q})$ (comment: radius of the initial ellipsoid/sphere E_0)
- $A_0 := R^2 I$
- $x_0 := 0$ (comment: center of E_0)
- $k := 0$

1. **Iterative Step:**

while $k < N$
call Strong Separation $(\mathcal{Q}, x^k)$
if $x^k \in \mathcal{Q}$ **halt**
else hyperplane $\{x \in \mathbb{R}^d \mid c^T x = c_0\}$ separates x^k from $\mathcal{Q}$
Update

$$b := \frac{1}{\sqrt{c^T A_k c}} A_k c$$
$$x^{k+1} := x^k - \frac{1}{d+1} b$$
$$A_{k+1} := \frac{d^2}{d^2-1}\left(A_k - \frac{2}{d+1} b b^T\right)$$
$$k := k + 1$$

endwhile

2. **Empty Polyhedron:**

- **halt** and **declare** "$\mathcal{Q}$ is empty"

3. **End**

The crux of the complexity analysis of the algorithm is on the a-priori determination of the iteration bound. This in turn depends on three factors- the volume of the initial ellipsoid E_0, the rate of volume shrinkage ($\frac{vol(E_{k+1})}{vol(E_k)} < e^{-\frac{1}{(2d)}}$) and the volume threshold at which we can safely conclude that $\mathcal{Q}$ must be empty. The assumption of $\mathcal{Q}$ being a rational polyhedron is used to argue that $\mathcal{Q}$ can be modified into a full-dimensional polytope without affecting the decision question ("Is $\mathcal{Q}$ nonempty?"). After careful accounting for all these technical details and some others (e.g., compensating for the round-off errors caused by the square root computation in the algorithm), it is possible to establish the following fundamental result:

Theorem 4.1 *There exists a polynomial $g(d, \phi)$ such that the* ellipsoid method *runs in time bounded by $T\, g(d, \phi)$ where ϕ is an upper bound on the size of linear inequalities in some description of $\mathcal{Q}$ and T is the maximum time required by the oracle* Strong Separation*$(\mathcal{Q}, y)$ on inputs y of size at most $g(d, \phi)$.*

The size of a linear inequality is just the length of the encoding of all the coefficients needed to describe the inequality. A direct implication of the theorem is that solvability of linear inequalities can be checked in polynomial time if strong separation can be solved in polynomial time. This implies that the standard linear programming solvability question has a polynomial-time algorithm (since separation can be effected by simply checking all the constraints). Happily, this approach provides polynomial-time algorithms for much more than just the standard case of linear programming solvability. The theorem can be extended to show that the optimisation of a linear objective function over $\mathcal{Q}$ also reduces to a polynomial

number of calls to the strong separation oracle on $\mathcal{Q}$. A converse to this theorem also holds, namely separation can be solved by a polynomial number of calls to a solvability/optimisation oracle (Grötschel et al. 1988). Thus, optimisation and separation are polynomially equivalent.

5 Karmarkar's interior point method

The announcement of the polynomial solvability of linear programming followed by the probabilistic analyses of the simplex method in the early 1980s left researchers in linear programming with a dilemma. We had one method that was good in a theoretical sense but poor in practice and another that was good in practice (and on average) but poor in a theoretical worst-case sense. This left the door wide open for a method that was good in both senses. Narendra Karmarkar closed this gap with a breathtaking new projective scaling algorithm. Many investigations followed to improve the implementations of the interior point method with so-called affine scaling variants of Karmarkar's algorithm. These were shown to be related to earlier work of Dikin. Researchers also noted how these affine scaling methods for linear programming could be interpreted as generalisations of the simplex method, Stone and Tovey (1991) and Chandru and Kochar (1985). Further work led to the successful predictor-corrector primal–dual methods which turned out to be the really practical codes for large scale problems. In retrospect, the new algorithm has been identified with a class of nonlinear programming methods known as logarithmic barrier methods. Implementations of a primal–dual variant of the logarithmic barrier method have proven to be the best approach at present. The monogragh by Wright (1997) is dedicated to primal–dual interior point methods. It is a variant of this method that we describe below.

It is well known that moving through the interior of the feasible region of linear program using the negative of the gradient of the objective function, as the movement direction, runs into trouble because of getting "jammed" into corners (in high dimensions, corners make up most of the interior of a polyhedron). This jamming can be overcome if the negative gradient is balanced with a "centering" direction. The centering direction in Karmarkar's algorithm is based on the **analytic center** y_c of a full dimensional polyhedron $\mathcal{D} = \{x : A^T y \leq c\}$ which is the unique optimal solution to

$$\max \left\{ \sum_{j=1}^{n} \ln(z_j) \; : \; A^T y + z = c \right\} .$$

Recall the primal and dual forms of a linear program may be taken as

$$\begin{aligned} (P) \quad & \min \{cx : Ax = b, \; x \geq 0\} \\ (D) \quad & \max \left\{ b^T y : A^T y \leq c \right\} . \end{aligned}$$

The logarithmic barrier formulation of the dual (D) is

$$(D_\mu) \quad \max \left\{ b^T y + \mu \sum_{j=1}^{n} \ln \left(z_j\right) : \; A^T y + z = c \right\} .$$

Notice that (D_μ) is equivalent to (D) as $\mu \to 0^+$. The optimality (Karush–Kuhn–Tucker) conditions for (D_μ) are given by

$$\begin{aligned} D_x D_z e &= \mu e \\ Ax &= b \\ A^T y + z &= c\,, \end{aligned}$$

where D_x and D_z denote $n \times n$ diagonal matrices whose diagonals are x and z, respectively. Notice that if we set μ to 0, the above conditions are precisely the primal–dual optimality conditions; complementary slackness, primal and dual feasibility of a pair of optimal (P) and (D) solutions. The problem has been reduced to solving the above equations in x, y and z. The classical technique for solving equations is Newton's method which prescribes the directions

$$\begin{aligned} \Delta y &= -\left(A D_x D_z^{-1} A^T\right)^{-1} A D_z^{-1} (\mu e - D_x D_z e) \\ \Delta z &= -A^T \Delta y \\ \Delta x &= D_z^{-1} (\mu e - D_x D_z e) - D_x D_z^{-1} \Delta z\,. \end{aligned} \tag{6}$$

The strategy is to take one Newton step, reduce μ, and iterate until the optimisation is complete. The criterion for stopping can be determined by checking for feasibility $(x, z \geq 0)$ and if the duality gap $(x^t z)$ is close enough to 0. We are now ready to describe the algorithm.

Procedure: **Primal-Dual Interior**

0. **Initialize:**

- $x_0 > 0\,,\ \ y_0 \in \mathbb{R}^m\,,\ \ z_0 > 0, \mu_0 > 0, \epsilon > 0, \rho > 0$
- $k := 0$

1. **Iterative Step:**

do
Stop if $Ax_k = b,\ \ A^T y_k + z_k = c$ and $x_k^T z_k \leq \epsilon$.
$x_{k+1} \leftarrow x_k + \alpha_k^P \Delta x_k$
$y_{k+1} \leftarrow y_k + \alpha_k^D \Delta y_k$
$z_{k+1} \leftarrow z_k + \alpha_k^D \Delta z_k$
/* Δx_k, Δy_k, Δz_k are the Newton directions from (6) */
$\mu_{k+1} \leftarrow \rho \mu_k$
$k := k + 1$
od

2. **End**

The primal–dual algorithm has been used in several large-scale implementations. For appropriate choice of parameters, it can be shown that the number of iterations in the worst-case is $O(\sqrt{n} \log(\epsilon_0/\epsilon))$ to reduce the duality gap from ϵ_0 to ϵ (Saigal 1995; Wright 1997). While this is sufficient to show that the algorithm is polynomial time, it has been observed that the "average" number of iterations is more like $O(\log n \log(\epsilon_0/\epsilon))$. However, unlike the simplex method we do not have a satisfactory theoretical analysis to explain this observed behavior.

The step sizes α_k^P and α_k^D are chosen to keep x_{k+1} and z_{k+1} strictly positive. The ability in the primal–dual scheme to choose separate step sizes for the primal and dual variables is

a major computational advantage that this method has over the pure primal or dual methods. Empirically this advantage translates to a significant reduction in the number of iterations.

The stopping condition essentially checks for primal and dual feasibility and near complementary slackness. Exact complementary slackness is not possible with interior solutions. When the algorithm terminates with an interior solution, a post-processing step is usually invoked to obtain optimal extreme point solutions for the primal and dual. This is usually called the purification of solutions and is based on a clever scheme, a folklore idea in mathematical programming, described by Megiddo (1991).

As in the case of the simplex method, there are a number of special structures in the matrix A that can be exploited by interior point methods to obtain improved efficiencies. Network flow constraints, generalised upper bounds (GUB), and variable upper bounds (VUB) are structures that often come up in practice and which can be useful in this context (Chandru and Kochar 1986; Todd 1986).

6 Integer programming on monotone inequalities

In a generalised network flow problem, each edge $e = (u, v)$ has a positive "flow multiplier" a_e associated with it, i.e. if a flow of x_e enters the edge at node u, then a flow of $a_e x_e$ exits the edge at v.

6.1 Uncapacitated generalised transshipment problem

The uncapacitated generalised transshipment problem is defined on a generalised network where demand and supplies are associated with the vertices and costs are associated with the edges. The aim is to find a flow such that the excess/deficit at each vertex equals the desired value of the supply/demand, and the sum over the edges of the product of the cost and the flow is minimised.

Given a generalised network, consisting of a graph $G = (V, E)$ and $in(i)$ ($out(i)$) denote the set of edges that go into (out) i, with flow multipliers a_e and edge-costs c_e ($e \in E$) and supplies/demands b_i, $i \in V$, we need to find a flow function $\mathbf{x} = (x_e)_{e \in E}$ to solve the following:

Minimize $\sum_{e \in E} c_e x_e$ subject to $\sum_{e \in in(i)} a_e x_e - \sum_{e \in out(i)} x_e = b_i$ ($i \in V$) and $x_e \geq 0$.

The dual can be stated as follows: to find $\pi_1, \ldots, \pi_n$ to solve:

Maximize $\sum_{i=1}^{n} b_i \pi_i$ subject to $\pi_i - a_e \pi_j \leq c_e$ ($e = (i, j) \in E$).

A matrix A is said to be a pre-Leontief matrix if A has at most one positive entry per column (Cottle and Veinott 1972). A system where, for every inequality, the two non-zero coefficients have opposite signs (both A^t and $-A^t$ are pre-Leontief) is called monotone. In a monotone system, if all the variables are bounded, then there exists a solution where all variables are maximised (or minimised) simultaneously.

The set of constraints of the dual of the uncapacitated generalised transshipment problem is monotone. Adler and Cosares (1991) reduced this problem, where only demand nodes (or only supply nodes) are present, to a single solution of a monotone system. Their algorithm is based on solving the dual linear programming problem. They showed that these instances can be solved by using one application of Megiddo's algorithm. Cohen and Megiddo's algorithms (1991, 1993, 1994) improve the running time and the uncapacitated generalised transshipment problem with only source nodes (demand nodes) can be solved in an expected running time

of $\mathcal{O}(n^3 \log n + mn \log m \log^3 n + mn \log^5 n)$ and a deterministic time of $\mathcal{O}(mn \log m + mn^2 \log^2 n)$.

The algorithms of Cohen and Megiddo (or the algorithm of Adler and Cosares) compute the simultaneous maximum or minimum point. Hence they do not solve the uncapacitated generalised transshipment problems where both supply and demand nodes are present.

6.2 Generalised circulation

Given a generalised network $G = (V, E)$ with a distinguished vertex s and capacities $c_{ij} \geq 0$ associated with edges, the generalised circulation problem is to find a feasible flow, i.e., $x_{ij} \geq 0$ which satisfies the flow conservation constraint $\sum_j x_{ij} - \sum_j a_{ji} x_{ji} = 0$ for $i \neq s$ and maximises the excess at s, defined as $\sum_j x_{sj} - \sum_j a_{js} x_{js}$.

The best known time bound for this problem is due to Vaidya (1989), which relies on theoretically fast matrix multiplication algorithms. The existence of a strongly polynomial algorithm for this problem is still an open question and is a problem of special interest because it is one of the simplest class of linear programming problems which does not yet have a strongly polynomial algorithm. Cohen and Megiddo's algorithm computes a feasible generalised flow whose excess approximates the maximum excess to any constant factor. This algorithm is based on iterative calls to an algorithm for monotone systems.

7 Conclusion

This primer on linear programming has hopefully conveyed to the reader a quick view of the historical drama of the field of linear optimisation as it played out in the second half of the twentieth century. It is significant that the advances in computing technology was a concurrent drama that, riding on Moore's law of semiconductor hardware, made the efficient solution of realistic models of many engineered and societal systems possible. Human progress based on these advances in fundamental automation has been the foundation on which we have now reached the second machine age where decision sciences are now creating ubiquitous machine intelligence applications at astonishing speed.

In today's optimisation software platforms like Gurobi Optimizer 7.0, the choice of solvers for linear programming is between the simplex methods and interior point methods. The characteristics of the two classes of solvers are a bit different as we have learned in this primer and the nature, the size and characteristics of the application will dictate the choice of algorithms to be invoked. It may not be far fetched to assume that there may be a machine learning based expert system that manages this choice of solvers in the near future.

Another future direction for the field may be the era of NoSQL databases that are rapidly gaining credence. The ability to run linear programming in these distributed and real-time paradigms of cyber physical systems would be a new chapter in a future primer.

Acknowledgements The authors would like to thank the anonymous reviewers for their detailed comments and suggestions for improvement of the original submission. The authors have incorporated most of their suggestions and of course take responsibility for any remaining flaws.

References

Adler, I., & Cosares, S. (1991). A strongly polynomial algorithm for a special class of linear programs. *Operations Research*, *39*, 955–960.

Bixby, R. E. (1994). Progress in linear programming. *ORSA Journal on Computing*, *6*(1), 15–22.

Borgwardt, K. H. (1987). *The simplex method: A probabilistic analysis*. Berlin: Springer.

Cerkinov, R. N. (1961). The solution of linear programming problems by elimination of unknowns. *Doklady Akademii Nauk*, *139*, 1314–1317.

Chandru, V., & Kochar, B. S. (1985). *A class of algorithms for linear programming*, Research Memorandum No. 85-14, Purdue University.

Chandru, V., & Kochar, B. S. (1986). *Exploiting special structures using a variant of Karmarkar's algorithm*, Research Memorandum No. 86-10, School of Industrial Engineering, Purdue University.

Chandru, V. (1993). Variable elimination in linear constraints. *The Computer Journal*, *36*(5), 463–472.

Charnes, A. (1952). Optimality and degeneracy in linear programming. *Econometrica*, *20*, 160–170.

Cohen, E., & Megiddo, N. (1991). Improved algorithms for linear inequalities with two variables per inequality. In *Proceedings of the twenty third symposium on theory of computing*, New Orleans (pp. 145–155).

Cohen, E., & Megiddo, N. (1993). New algorithms for generalized network flows, Revised. In D. Dolev, Z. Galil, & M. Rodeh (Eds.), *Proceedings of the 1st Israeli symposium on the theory of computing and systems* (pp. 103–114).

Cohen, E., & Megiddo, N. (1994). Improved algorithms for linear inequalities with two variables per inequality, Extended Abstract. *SIAM Journal of Computing*, *23*, 1313–1347.

Cottle, R. W., & Veinott, A. F, Jr. (1972). Polyhedral sets having a least element. *Mathematical Programming*, *3*, 238–249.

Dantzig, G. B. (1951). Maximization of a linear function of variables subject to linear inequalities. In C. Koopmans (Ed.), *Activity analysis of production and allocation* (pp. 339–347). New York: Wiley.

Dantzig, G. B. (1963). *Linear programming and extensions*. Princeton: Princeton University Press.

Dantzig, G. B., & Eaves, B. C. (1973). Fourier–Motzkin elimination and its dual. *Journal of Combinatorial Theory (A)*, *14*, 288–297.

Dantzig, G. B., Orden, A., & Wolfe, P. (1955). The generalized simplex method for minimizing a linear form under linear inequality restraints. *Pacific Journal of Mathematics*, *5*, 183–195.

Fourier, L. B. J. (1826). *Reported in: Analyse des travaux de l'Academic Royale des Sciences. pendant l'annee* (1823), Partie mathematique, Histoire de l'Academie Royale des Sciences de l'Institut de France (Vol. 6, pp. xxix–xli).

Fourier, L. B. J. (1827). *Reported in: Analyse des travaux de l'Academic Royale des Sciences. pendant l'annee* (1824), Partie mathematique, Histoire de l'Academie Royale des Sciences de l'Institut de France (Vol. 7, pp. xlviii–lv) (Partial English Translation in: D.A. Kohler, Translation of a report by Fourier on his work on linear inequalities. Opsearch, Vol. 10, pp. 38–42, 1973)

Grötschel, M., Lovász, L., & Schrijver, A. (1982). The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, *1*, 169–197.

Grötschel, M., Lovász, L., & Schrijver, A. (1988). *Geometric algorithms and combinatorial optimization*. Berlin: Springer.

Hačijan, L. G. (1979). A polynomial algorithm in linear programming. *Soviet Mathematics Doklady*, *20*, 191–194.

Haimovich, M. (1983). *The simplex method is very good! On the expected number of pivot steps and related properties of random linear programs*. Unpublished Manuscript.

Hochbaum, D. S., & Naor, J. (1994). Simple and Fast Algorithms for Linear and Integer Programs with two variables per inequality. *SIAM Journal on Computing*, *23*(6), 1179–1192.

Karmarkar, N. K. (1984). A new polynomial-time algorithm for linear programming. *Combinatorica*, *4*, 373–395.

Karp, R. M., & Papadimitriou, C. H. (1982). On linear characterizations of combinatorial optimization problems. *SIAM Journal on Computing*, *11*, 620–632.

Klee, V., & Minty, G. J. (1972). How good is the simplex algorithm? In O. Shisha (Ed.), *Inequalities III*. Cambridge: Academic Press.

Lassez, J.-L. (1991). From LP to LP: Programming with constraints. In *Proceedings of theoretical aspects of computer software*, Sendai.

Lassez, J.-L., & Maher, M. J. (1988). *On Fourier's algorithm for linear arithmetic constraints*, IBM Research Report, T Watson Research Center.

Megiddo, N. (1983). Towards a genuinely polynomial algorithm for linear programming. *SIAM Journal on Computing*, *12*(2), 347–353.

Megiddo, N. (1991). On finding primal- and dual-optimal bases. *ORSA Journal on Computing*, *3*, 63–65.

Motzkin, T. S. (1936). *Beitrage zur theorie der linearen Ungleichungen*. Doctoral thesis, University of Base.
Padberg, M. W., & Rao, M. R. (1981). *The Russian method for linear inequalities, Part III, Bounded integer programming*. New York: New York University. (preprint).
Saigal, R. (1995). *Linear programming: A modern integrated analysis*. Alphen aan den Rijn: Kluwer Press.
Schrijver, A. (1986). *Theory of linear and integer programming*. Hoboken: Wiley.
Shor, N. Z. (1970). Convergence rate of the gradient descent method with dilation of the space. *Cybernetics*, *6*, 102–108.
Stone, Richard E., & Tovey, Craig A. (1991). The simplex and projective scaling algorithms as iteratively reweighted least squares methods. *SIAM Review*, *33*(2), 220–237.
Todd, M. J. (1986). *Exploiting special structure in Karmarkar's algorithm for linear programming*. Technical Report 707, School of Operations Research and Industrial Engineering, Cornell University.
Todd, M. J. (1994). Theory and practice for interior-point methods. *ORSA Journal on Computing*, *6*(1), 28–31.
Vaidya, P. M. (1989). Speeding-up linear programming using fast matrix multiplication. In *Proceedings of the 30th IEEE annual symposium on foundations of computer science* (pp. 332–337).
Vanderbei, R. J. (2015). *Linear programming: Foundations and extensions* (4th ed.). Berlin: Springer.
Williams, H. P. (1986). Fourier's method of linear programming and its dual. *The America Mathematical Monthly*, *93*, 681–695.
Wolfe, P. (1963). A technique for resolving degeneracy in linear programming. *SIAM Journal on Applied Mathematics*, *11*(205), 211.
Wright, S. J. (1997). *Primal-dual interior-point methods*. Cambridge: SIAM Press.
Ziegler, M. (1995). *Convex polytopes*. Berlin: Springer.