Available online at www.sciencedirect.com
ScienceDirect
Discrete Optimization 5 (2008) 174–185
DISCRETE OPTIMIZATION
www.elsevier.com/locate/disopt

George Dantzig's impact on the theory of computation

Richard M. Karp*

University of California at Berkeley, Berkeley, CA, United States
International Computer Science Institute, Berkeley, CA, United States

Received 19 December 2005; accepted 18 December 2006
Available online 26 December 2007

Abstract

George Dantzig created the simplex algorithm for linear programming, perhaps the most important algorithm developed in the 20th century. This paper traces a single historical thread: Dantzig's work on linear programming and its application and extension to combinatorial optimization, and the investigations it has stimulated about the performance of the simplex algorithm and the intrinsic complexity of linear programming and combinatorial optimization.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Linear programming; Simplex algorithm; Combinatorial optimization; Integer programming; Computational complexity; Polynomial-time algorithm

1. Dantzig's great contributions

George Dantzig will go down in history as one of the founders and chief contributors to the field of mathematical programming, and as the creator of the simplex algorithm for linear programming, perhaps the most important algorithm developed in the 20th century.

Dantzig's formulation of linear programming as a broadly applicable tool for solving planning problems grew out of his experience as a practical program planner for the US Air Force during and after World War II, and his recognition that computers were emerging as a tool with vast potential for solving problems of planning and resource allocation. The following quote illuminates Dantzig's perspective on this work.

"Linear programming can be viewed as part of a great revolutionary development which has given mankind the ability to state general goals and to lay out a path of detailed decisions to take in order to 'best' achieve its goals when faced with practical situations of great complexity. Our tools for doing this are ways to formulate real-world problems in detailed mathematical terms (models), techniques for solving these models (algorithms), and engines for executing the steps of algorithms (computers and software)."

The noted combinatorial mathematician Alexander Schrijver has stated that "Linear programming forms the hinge in the history of combinatorial optimization". Indeed, many combinatorial optimization problems can be solved by

* Corresponding address: Department of Electrical Engineering and Computer Sciences, Computer Science Division, Soda Hall, University of California, Berkeley, CA 94720, United States.
E-mail address: karp@icsi.berkeley.edu.

1572-5286/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.disopt.2006.12.004

representing them as linear programming problems, although formulating the right system of linear constraints to achieve this representation can be a very subtle matter. Dantzig's early work paved the way for this powerful approach to combinatorial optimization. Although the field of combinatorial optimization has branched out in many directions, including efficient algorithms for problems with special structure, cutting-plane methods, branch-and-cut methods, *NP*-completeness, approximation algorithms for *NP*-hard optimization problems, and theoretical results on hardness of approximation, the structure of the entire field was established by Dantzig and the generation of superb researchers who developed the foundations of linear and nonlinear programming and network flow theory in the 1950s and early 1960s at research centers such as the Rand Corporation, Princeton University, the National Bureau of Standards and IBM Research. My own lifelong interest in this field began with the opportunity to meet such giants as Dantzig, Merrill Flood, Ray Fulkerson, Ralph Gomory, Alan Hoffman, Harold Kuhn and Albert Tucker at an IBM symposium in 1959, near the beginning of my research career.

This paper follows a single historical thread: Dantzig's work on linear programming and its application and extension to combinatorial optimization, and the investigations it has stimulated about the performance of the simplex algorithm, the intrinsic complexity of linear programming and combinatorial optimization, and the relevance of worst-case performance as a measure of algorithmic efficiency.

For an overview of Dantzig's achievements in mathematical programming up to 1963 the reader is referred to his classic text "Linear Programming and Extensions" [18]. For a more rounded account of Dantzig's contributions to areas such as large-scale linear programming, economic modeling, linear programming under uncertainty, nonlinear programming and complementarity problems see "The Basic George B. Dantzig", Richard Cottle's excellent collection of selected papers by Dantzig [14].

2. Linear programming, the simplex algorithm and combinatorial optimization

Linear programming is the problem of minimizing a linear function subject to linear inequality constraints. It can be stated algebraically as follows: Minimize $c \cdot x$ subject to $Ax \geq b$ and $x \geq 0$. The data for the problem consists of the $n \times d$ matrix A, the d-vector c and the n-vector b. The components of the d-vector x are the decision variables of the problem. In planning problems the variables typically represent the amount of various items produced or consumed and the linear inequalities represent constraints on the production or consumption processes, but, in the diverse applications of linear programming, many other interpretations of the variables and constraints may arise.

Associated with every linear program is a dual linear program. The dual of the linear program $\min c \cdot x$ subject to $Ax \geq b$ and $x \geq 0$ is the linear program $\max u \cdot b$ subject to $u'A \leq c$ and $u \geq 0$. The duality theorem, which Dantzig first heard of in a famous discussion with John von Neumann, states, among other things, that if a linear program and its dual both have feasible solutions then both have optimal solutions, and the minimum value of $c \cdot x$ is equal to the maximum value of $u \cdot b$.

Geometrically, the linear programming problem is to find the furthest point in a given direction on a convex polyhedron (the intersection of a finite number of half-spaces). If the polyhedron is nonempty and bounded then an optimal solution exists and occurs at a vertex (extreme point) of the polyhedron. Starting at an initial extreme point the simplex algorithm follows edges of the polyhedron, always moving in the desired direction, until an extreme point is reached where no edge allows progress in the desired direction. Subject to a certain nondegeneracy condition, such an extreme point provides an optimal solution. Each transition along an edge is called a pivot step. If an initial extreme point is not available then one can be found by a so-called Phase I procedure which introduces additional variables to create an artificial linear programming instance in which an initial extreme point is available and an optimal solution, which can be found by the simplex method itself, yields the desired initial extreme point for the original problem. The simplex method uses the Phase I procedure to find an extreme point of the polyhedron, followed by additional pivot steps to find an optimal extreme point.

The simplex algorithm is possibly the most important algorithmic development of the 20th century. Several refined modeling languages and software implementations of different variants of the simplex algorithm are widely used, and they are consistently efficient in solving problems with tens of thousands or more of variables and constraints. The simplex algorithm poses a dilemma to the theory of computation because, in contrast to its superb performance in practice, pathological families of examples have been constructed showing that several of the principal variants of the simplex algorithm can be made to run for an exponential number of steps. Thus the immensely successful algorithm

on which the entire field of mathematical programming is based lacks a pedigree according to the main yardsticks of the theory of computation, which measure an algorithm by its worst-case performance.

George Dantzig's approach to optimization problems was fundamentally pragmatic. Despite the pathological examples he knew that the simplex algorithm was an incomparably effective tool, and he bent his energies to modeling real applications, producing improved software tools, and developing algorithms for special cases of linear programming and extensions to integer, nonlinear and stochastic programming.

In this paper we describe some of the efforts of workers in the theory of computation to explain the success of the simplex algorithm and explore the intrinsic complexity of linear programming. We will describe work on the probabilistic analysis of versions of the simplex algorithm. We will present a very recent result showing that an algorithm closely resembling the simplex algorithm solves the linear programming problem in randomized polynomial time. We will also describe a recent attempt to develop a variant of the simplex algorithm which is *strongly polynomial* – *i.e.*, terminating within a number of steps bounded by a polynomial in n and d. Although Dantzig's orientation was deeply practical, he had a strong interest in these lines of theoretical investigation and made some effort to contribute to them. We will also describe some linear programming algorithms that provably run in deterministic polynomial time: the ellipsoid algorithm, which is primarily of theoretical interest, and the interior-point methods, which are worthy competitors with the simplex algorithm in practice.

Integer programming is the problem of minimizing a linear function subject to linear constraints, with the additional requirement that the decision variables be integer-valued. As we shall describe, Dantzig was among the first to recognize the universal character of integer programming as a format for describing combinatorial problems. We will describe his pioneering work showing that certain integer programming problems are solvable by linear programming, since the integrality constraints will automatically be satisfied, as well as his early work on cutting-plane methods, in which integer programming problems are solved by introducing linear constraints to eliminate fractional solutions. We will go on to describe a major theme in combinatorial optimization, in which a problem which seems to require integer programming can be formulated as a linear program by implicitly considering a possibly exponential number of linear constraints inherent in the combinatorial structure of the problem, and obtaining solutions via the primal–dual method, a simplex variant due to Dantzig, Ford and Fulkerson.

3. The theoretical efficiency of variants of the simplex method

From a practical point of view the efficiency of the simplex method is beyond doubt. The most efficient simplex codes, such as CPLEX, typically solve linear programs with tens of thousands of variables and constraints, or more, in a few minutes, with a number of pivot steps that is roughly linear in $\min(d, n)$, where d is the number of variables and n is the number of constraints. On the other hand, for several of the common pivot rules, families of instances have been constructed for which the number of pivot steps grows exponentially with $\min(d, n)$. Thus, from the point of view of worst-case performance, these variants of the simplex algorithm are not efficient. The first of these constructions was due to Klee and Minty [35] and a unified approach to constructing such families of examples is given in [8]. If a randomized pivot rule is allowed then the situation is a bit more favorable: A randomized pivot rule for which the expected number of pivots is $n^{O(d)}$ is given in [29].

Despite these negative results there have been many attempts to establish favorable theoretical properties of some version of the simplex algorithm. The three principal directions have been probabilistic analysis, randomized algorithms and the (thus far unsuccessful) search for a strongly polynomial variant.

3.1. Probabilistic analysis

Borgwardt [11,12], Smale [41] and Megiddo [38] showed that, for certain simplex variants, the expected number of pivot steps is bounded by a polynomial in n and d when the instances are drawn from a spherically symmetric distribution. Haimovich [27] and Adler [1] initiated a line of research, further refined in [44,6,5] showing that, for any constraint matrix A, the expected number of pivot steps for a parametric simplex algorithm is bounded above by a quadratic function of n and d when the directions of the inequalities are chosen at random. However, it is not clear that results proved under the assumptions of spherical symmetry or random directions for the inequalities have much power to explain the performance of the simplex algorithm on real-life instances.

3.2. Smoothed analysis

Recently the concept of *smoothed analysis* of algorithms was introduced and applied to linear programming in [42], giving a somewhat more satisfying insight into the favorable performance of the simplex algorithm. The paper [42] considers a two-phase simplex algorithm based on a shadow vertex pivot rule. The shadow vertex algorithm is defined as follows. Suppose we are given a vertex x^* of the polyhedron $Q = \{x|Ax \geq b\}$. Let t be a d-vector such that x^* is an optimal solution of the linear program $\min t \cdot x$ subject to $x \in Q$. Define the (c, t)-*shadow* S of the polyhedron Q as the projection of Q onto the two-dimensional space spanned by the vectors t and c. Let x_{opt} be an optimal extreme point solution of: $\min c \cdot x$ such that $x \in Q$. The boundary of the shadow S contains a path from the image of x^* to the image of x_{opt}, and the shadow vertex algorithm passes through the preimages in Q of the vertices along this path in S. This path along the edges of Q can be computed by suitable pivot steps without explicitly constructing the polygon S. The two-phase shadow vertex algorithm applies this method first to a Phase I linear program using artificial variables to obtain a vertex of Q, and then to the polyhedron Q.

The main result of [42] states that if we take any linear programming instance $\min c \cdot x$ subject to $Ax \geq b$ with d variables and n constraints, and then perturb each entry of A and b independently by adding a Gaussian random variable of mean 0 and standard deviation σL, where a_i is the ith row of A, b_i is the ith component of b, and L is the maximum Euclidean length of any vector (a_i, b_i), then the expected number of pivot steps required to solve the resulting perturbed linear program will be bounded by a certain polynomial $P(n, d, \frac{1}{\sigma})$.

The implications of this result are clearest for applications in which the data arise from some measurement or estimation process and are inherently noisy; in such cases one may think of the estimation process as inherently providing the required random perturbation of the true values of the data. Even there, it is troubling that the standard deviation of the perturbations needs to be large to ensure a reasonable running time, and is the same for all elements A and b, whether they are large or small. Moreover, the analysis has little meaning in the applications of linear programming to combinatorial problems, where A may contain a zero-one submatrix representing a discrete structure such as the node-arc incidence matrix of a graph; small perturbations of the matrix in such cases may destroy the meaning of the linear programming model. It would be of interest to extend the theorem to perturbations that alter some of the data while leaving the combinatorial structure unchanged.

3.3. A randomized polynomial-time simplex-like algorithm

The paper [33] gives a randomized polynomial-time algorithm based on simplex-like pivot steps which, on any linear programing instance possessing an optimal solution, produces an optimal solution with probability at least 3/4. We shall give a brief sketch of this algorithm, neglecting several essential technical details.

The algorithm is based on a reduction from the linear programming problem to the problem of determining whether a polytope defined by linear inequalities is bounded, together with the use of the shadow-vertex simplex method to test boundedness in polynomial time (with high probability).

The linear program is in the form: Maximize $c \cdot x$ subject to $Ax \leq b$.

Assuming that this program and its dual are both feasible and bounded, x and y are optimal solutions for the primal and dual respectively if and only if they satisfy the following system of inequalities and equalities:

$$Ax \leq b, \qquad A^T y = c, \quad y \geq 0 \quad \text{and} \quad c \cdot x = b \cdot y.$$

In polynomial time one can construct matrices A_1 and A_2 such that the following are equivalent:

(i) The linear program is feasible;
(ii) The zero vector is in the convex hull of the rows of A_1;
(iii) The zero vector is in the interior of the convex hull of the rows of A_2;
(iv) The polytope $\{w|A_2 w \leq b_1\}$ is bounded, where b_1 is an arbitrary positive vector.

A_1 is constructed from the linear program by simple transformations: replacing each variable by the difference of two nonnegative variables, introducing slack variables and rescaling, producing the homogeneous system $A_1 z = 0$, $z \geq 0$, $z \neq 0$, whose feasibility is equivalent to (ii). A_2 is constructed from A_1 by applying a small perturbation to avoid the degenerate case in which the zero vector is on the boundary of the convex hull of the rows of A_1. The equivalence of (iii) and (iv) is a generic property, holding for any matrix A_2.

One can prove (iv) by exhibiting vertices that maximize and minimize some linear objective function. Moreover, from these vertices one can backtrace in polynomial time through the constructions of A_2 and A_1 to obtain an optimal solution to the original linear program.

To prove (iv), the algorithm picks a random positive vector b_1 and a random linear objective function $d \cdot z$ and runs the shadow-vertex algorithm, over a 2-dimensional subspace containing d, for a polynomial-bounded number of steps. It can be shown that, with probability at least $1/2$, the algorithm will either find vertices maximizing and minimizing $d \cdot z$, thus proving (iv), or will reach a vertex of large norm. In the latter case, the right-hand side b_1 is suitably modified and the shadow-vertex algorithm is repeated. It can be shown that, with high probability, a proof of (iv) will be obtained within polynomially many iterations if (iv) holds.

This algorithm runs in time polynomial in the number of bits needed to specify the linear program, but not in strongly polynomial time.

3.4. Towards a strongly polynomial simplex algorithm

Computational complexity theorists define a *polynomial-time algorithm* as one in which the number of steps to solve any instance is bounded by a polynomial function of the number of bits needed to represent the instance. An algorithm is called *strongly polynomial* if its input is an array of rational numbers, the only operations allowed are arithmetic operations and comparisons, the number of steps required to solve any instance is a polynomial function of the dimension of the array, and the lengths in bits of all intermediate results are polynomial in the length of the input. For example, the Gaussian elimination algorithm for inverting a $n \times n$ matrix is strongly polynomial, because the input contains n^2 numbers and the number of operations is $O(n^3)$. A strongly polynomial algorithm for the minimum-cost flow problem, a special case of the linear programming problem, is given in [43].

While practical implementations of the simplex algorithm may involve many details of finite precision and numerical roundoff, it is attractive to take a more algebraic view in which the input consists of rational numbers and the algorithm is restricted to exact arithmetic operations and comparisons. Moreover, if one could show in this algebraic framework that some simplex variant was strongly polynomial, it would not be difficult to transfer the analysis to a finite precision model, yielding a polynomial-time algorithm. For this reason, Dantzig and others expended considerable effort on trying to find a strongly polynomial variant of the simplex algorithm. To achieve this it would be sufficient to find a variant in which the number of pivot steps is bounded by a polynomial in n.

3.4.1. The Hirsch conjecture and abstract polytopes

Consider the simplex algorithm in the case where the feasible set $Q = \{x | Ax \geq b, x \geq 0\}$ is a bounded polyhedron in d-space defined by $n + d$ constraints, and an initial vertex v is given. To solve this instance the simplex algorithm must perform pivots corresponding to the successive edges along a path from the start vertex to the optimal vertex, and the number of pivots will be the length of this path. Define the *diameter* of a bounded polyhedron as the maximum, over all pairs (s, t) of vertices, of the minimum number of edges in a path from s to t. Then there cannot exist a pivot rule yielding a strongly polynomial algorithm unless the diameter of any bounded polyhedron is no greater than some polynomial-bounded function of n and d.

The well-known Hirsch conjecture [18] states that a bounded polyhedron in R^d defined by n constraints has diameter at most $n - d$. This conjecture remains open, and the best upper bound on diameter currently known is $n^{\log_2 d+2}$, due to Kalai and Kleitman [30].

Dantzig's keen interest in the Hirsch conjecture led to his formulation of the concept of an *abstract polytope* and his investigation of this concept with Ilan Adler, one of his first Ph.D. students at Stanford [2,3]. An abstract polytope is defined by a universe of $n + d$ elements and a graph in which each vertex is labeled with a distinct set of d elements from the universe, such that:

1. If two vertices are adjacent then the intersection of their label sets has cardinality $d - 1$;
2. If two vertices have label sets S and T then they are joined by a path in which the label set of each vertex is a subset of $S \cup T$.

Under a nondegeneracy assumption we can associate an abstract polytope with a bounded polyhedron $Q = \{x | Ax \geq b, x \geq 0\}$ where A is $n \times d$. The graph of the abstract polytope is the graph determined by the vertices and edges of Q, the universe is the set of $n + d$ linear inequalities, and the label of each vertex is the set of inequalities that hold as equalities at that vertex.

Since every nondegenerate inequality system determines an abstract polytope, an upper bound on the diameter of the graph of an abstract polytope would imply the same upper bound on the diameter of a polytope. However, although they established many interesting properties of abstract polytopes, Adler and Dantzig did not succeed in establishing an upper bound on diameter that was polynomial in n and d.

3.4.2. Achieving polynomial diameter

Koltun [36] suggested an approach that gets around the potential difficulty that the diameter may not be polynomial-bounded. The approach considers the *feasibility problem* of testing whether a system $Ax \geq b$ of n linear inequalities in d nonnegative variables has a solution. A strongly polynomial algorithm for the feasibility problem would imply a strongly polynomial algorithm for the optimization version of the problem.

Koltun's *arrangement method* transforms a linear programming feasibility instance with d variables and n constraints to an optimization instance with $d+1$ variables and n constraints on a bounded polyhedron with diameter at most $d(n-2d+1)$, thus eliminating the potential obstacle of super-polynomial diameter.

Koltun's original construction is rather elaborate. Two recent working papers achieve the same properties with more natural constructions. Hazan and Megiddo [28] consider the standard "Phase I" technique of recasting the feasibility problem as an optimization problem: Minimize $e^T s$ subject to $Ax+s \geq b, s \geq 0$, where e is a vector of 1's. Adler and Koltun [4] take the feasibility problem in the form $Ax = b, x \geq 0$ and recast it as the following optimization problem: Minimize $e^T x^-$ subject to $Ax^+ - Ax^- = b, x^+ \geq 0, x^- \geq 0$. In each case, the authors show that the polyhedron in their optimization problem has a polynomial-bounded diameter.

These results imply that the solution of a linear programming feasibility problem is polynomial-time reducible to the solution of a linear programming optimization problem on a polytope with polynomial-bounded diameter. Although these results remove the obstacle of a potentially superpolynomial-size diameter, they leave open the challenge of finding a strongly polynomial version of the simplex algorithm.

4. Linear programming lies in *P*

The efficiency of several different variants of the simplex method, contrasted with their exponential running time on artificially constructed instances, stimulated great interest in the question of whether the linear programming problem is solvable in polynomial time by a deterministic algorithm. This question was resolved affirmatively by Khachiyan in 1979 [34].

4.1. Linear programming lies in $NP \cap co-NP$

Until Khaciyan proved his result the linear programming feasibility problem had an anomalous status from the point of view of computational complexity theory because it lay in the intersection of the complexity classes *NP* and $co-NP$ and yet was not known to lie in P. To explain this further, let us recall some of the concepts of complexity theory. A (decision) problem is defined as a subset L of $\{0,1\}^*$, the set of finite strings over the alphabet $\{0,1\}$. For example, the problem of recognizing composite numbers is defined as the set of binary representations of composite numbers. A problem L lies in P if there is a deterministic polynomial-time algorithm for recognizing the strings in L. L lies in *NP* (nondeterministic polynomial time) if there is a polynomial-time verification algorithm which accepts pairs (x, w), where x is a string and w is a string of length polynomial in the length of x, such that the following property holds: a string x lies in L if and only if there exists a string w such that the polynomial-time verifier accepts (x, w). For example, if L is the set of binary representations of composite numbers, then string x lies in L if and only if there is a string w such that the number represented by w is a proper divisor of the number represented by x. Since this property of the pair (x, w) can be checked in polynomial time, the set of composite numbers lies in *NP*. The central problem of complexity theory is whether $P = NP$. Finally, L lies in $co-NP$ if and only if the complementary set $\{0,1\}^* - L$ lies in *NP*.

It is immediate that P is contained in $NP \cap co-NP$. Complexity researchers tend to conjecture that P is a proper subset of $NP \cap co-NP$, although no proof is known, and a proof would settle in the negative the open question of whether $P = NP$. However, very few natural examples are known of problems that lie in $NP \cap co-NP$, but have not been shown to lie in P. For some time the most prominent candidates were the linear programming feasibilty problem and the problem of deciding whether a number is prime, which was proven to lie in P in 2002 [7].

We can define the linear programming feasibility problem as the set of pairs (A, b) such that $Ax = b$ has a nonnegative solution (the use of equalities instead of inequalities in the presence of nonnegativity constraints does not affect the complexity of the problem). The fact that this problem lies in both *NP* and $co-NP$ follows from the *Farkas lemma*, which states that, for any $m \times n$ matrix A and m-vector b, exactly one of the following holds:

1. There exists a nonnegative vector x such that $Ax = b$;
2. There exists a vector p such that $p'A \geq 0$ and $p'b < 0$.

Moreover, the length of the binary representation of x in case (1) and p in case (2) is bounded by a polynomial in the length of the binary encoding of the pair (A, b). Thus the Farkas lemma implies that this problem lies in both *NP*, with x as a witness of feasibility, and in $co - NP$, with p as a witness of infeasibility.

4.2. The ellipsoid algorithm

Although it was strongly suspected that linear programming feasibility lies in P, no proof was known until 1979, when Khaciyan presented his ellipsoid algorithm. An *ellipsoid* in R^d is the image of the unit ball under a positive definite linear transformation. Algebraically, it is defined as the set of points in R^d satisfying $(x-z)'D^{-1}(x-z) \leq 1$, where D is a positive definite $d \times d$ matrix and the point z is called the *center* of the ellipsoid.The algorithm is based on the geometric fact that, in R^d, if E is an ellipsoid and H is a halfspace with the center of the ellipsoid on its boundary, then $E \cap H$ is contained in an ellipsoid F such that $vol(F) \leq \exp(\frac{-1}{2d+1})Vol(E)$, and a specification of F can be computed efficiently from a specification of E.

Now suppose we wish to decide whether $Q = \{x|Ax \geq b\}$ is nonempty. We assume for simplicity that we are given A, b and positive numbers V and v such that Q is either empty or has d-dimensional volume at least v, and Q is contained in a ball E_0 of volume V centered at the origin. Then, starting with E_0, the algorithm constructs a sequence of ellipsoids containing Q of smaller and smaller volume by repeating the following step: If the center of the current ellipsoid E lies in Q then declare that Q is nonempty and halt. Otherwise, one of the linear inequalities defining Q is violated at the center. This inequality defines a hyperplane separating the center from Q, and the next ellipsoid in the sequence contains the intersection of E with the halfspace consisting of those points separated from the center by the hyperplane. The process continues until either a feasible point is determined or the volume of the ellipsoid drops below v, implying that Q must be empty, since, if Q were nonempty, any ellipsoid containing Q would have volume at least v.

It can be shown that the number of iterations of the ellipsoid method is $O(L^6)$ where L is the number of bits required to specify the matrix A and vector b. It follows that the ellipsoid method is a polynomial-time algorithm for the linear programming feasibility problem, and it can also be extended to a polynomial-time algorithm for solving the optimization problem of minimizing $c \cdot x$ subject to the system of linear inequalities $Ax \geq b$.

In practice, the ellipsoid method is far inferior to the simplex algorithm and has little practical utility, since its typical number of iterations is close to the worst-case number. It does have the attractive theoretical property that it yields a polynomial-time algorithm for any optimization problem of the form $\min c \cdot x$ subject to $x \in Q$, where Q is a convex body in R^d for which the following separation problem can be solved in polynomial time: given a point $x \in R^d$, either determine that $x \in Q$ or find a hyperplane separating x from Q. This property of the ellipsoid algorithm has been a powerful tool for showing that particular combinatorial optimization problems are solvable in polynomial time. The ellipsoid method has also been extended to solve general semidefinite programs and convex quadratic programs.

4.3. Interior point algorithms

In contrast to the simplex algorithm, which follows the edges of the polyhedron Q, interior point methods follow a path through the interior of Q. Starting with Karmarkar's seminal 1984 paper [31] interior point methods were developed that run in polynomial time and rival or sometimes even surpass the efficiency of the simplex algorithm in practice. These methods start with a feasible point and follow a path that seeks to decrease the objective function while staying away from the boundary of Q. A good overview of these methods is given in [10]. Today's software libraries for linear programming often include implementations of both the simplex algorithm and an interior point algorithm. Both types of algorithms continue to be immensely valuable in practice, and there is no clear criterion for choosing one algorithm over the other in a given application.

5. Dantzig's contributions to combinatorial optimization

Integer programming is the problem of minimizing a linear function subject to linear inequality constraints and the requirement that the variables assume integer values. Nearly all combinatorial optimization problems of interest can be expressed naturally as integer programs in which each variable is constrained to be zero or one. In principle any integer program can be converted to a linear program by adjoining a finite number of linear inequality constraints that are not included in the original constraints of the integer program but are implied by those constraints. Such additional constraints are called *cuts*. In 1958 Gomory [26] gave a general method for solving any integer programming problem in a finite number of steps by starting with the given linear inequality constraints and generating cuts in the course of executing the simplex algorithm.

We describe three aspects of the interplay between linear programming and integer programming:

1. The identification by Dantzig and others of classes of integer programming problems in which the integrality constraints can be discarded, since the extreme point solutions of the linear inequality constraints are guaranteed to be zero-one vectors; such problems can be solved efficiently as linear programs.

   More generally, the *integrality gap* of an integer program is the difference between its optimal value and the optimal value of its linear programming relaxation, which is obtained by discarding the integrality constraints, thus admitting fractional solutions. For many combinatorial optimization problems expressed as integer programs, it can be proven that the integrality gap is small, and a provably near-optimal solution can be constructed from the linear programming relaxation, either by rounding the optimal fractional solution or by using a modification of the primal–dual method to obtain a feasible integer solution to the primal and an optimal solution to the dual, and showing that their values are close. This approach yields polynomial-time *approximation algorithms* guaranteed to provide near-optimal solutions for a wide class of *NP*-hard combinatorial optimization problems. An excellent treatment of this approach is given in [45].
2. A seminal 1954 paper by Dantzig, Fulkerson and Johnson [22] in which a 49-city traveling-salesman problem is solved by a *cutting-plane method* which adds judiciously chosen cuts in the course of executing the simplex algorithm.
3. A powerful approach, pioneered by Jack Edmonds, for solving combinatorial problems with a special structure. The key idea is to express the given problem as a linear program with a very large but systematically defined set of constraints. These constraints can be handled implicitly within the primal–dual method (a variant of the simplex algorithm invented by Dantzig, Ford and Fulkerson [20]), yielding a polynomial-time algorithm. In addition, comparison of the implicitly defined linear program and its dual yields an interesting theoretical characterization of the optimal solution.

5.1. Linear programs with integer extreme points

The general form of a zero-one integer program is: $\min c \cdot x$ subject to the linear constraints $Ax \geq b$ and the integrality constraints $x \in \{0, 1\}^d$. Many of the landmark results in combinatorial optimization show that, for particular problems, the extreme points of the the system of linear constraints also satisfy the integrality constraints. In such cases the integrality constraints can be dropped and the problem can be solved as a linear program.

One of the first results along these lines is due to Dantzig [15], who observed that the assignment problem has this special structure. The assignment problem can be stated as follows: given n workers and n jobs, let c_{ij} be the cost of assigning worker i to job j; find a one-to-one assignment of workers to jobs that minimizes total cost. The integer programming formulation is:

$\min \sum_i \sum_j c_{ij} x_{ij}$ subject to: for all i and j, $x_{ij} \geq 0$; for all j, $\sum_i x_{ij} = 1$; for all i, $\sum_j x_{ij} = 1$; and the integrality of the x_{ij}, with the interpretation that $x_{ij} = 1$ if worker i is assigned to job j. Dantzig noted that, by a theorem of Birkhoff, the extreme points of the system of linear inequalities are zero-one vectors representing the one-to-one assignments of workers to jobs, so the explicit integrality constraints can be dropped and the problem can be solved as a linear program.

This result was the precursor of a major stream of research identifying classes of zero-one integer programs that are naturally presented as integer programs, but can be solved as linear programs, since the extreme points of the system of linear constraints are integral.

In a 1956 paper [21] Dantzig and Fulkerson showed that the max-flow problem with integer capacities can be cast as a linear program with integral extreme points. Application of the duality theorem to this program yields a

proof of the max-flow min-cut theorem. Similar results hold for the min-cost flow problem and for many other basic optimization problems. In the same vein, Dantzig formulated the shortest-path problem as a linear program, and gave a combinatorial description of the application of the simplex algorithm to this problem; however, the resulting shortest-path algorithm turned out to require exponential time in the worst case. Additional contributions of Dantzig to shortest-path algorithms are given in [16,19].

5.2. The traveling-salesman problem

In [22] Dantzig, Fulkerson and Johnson initiated the study of cutting-plane methods by solving a 49-city instance of the traveling-salesman problem: given n cities and the cost of traveling from any city to any other, find a minimum-cost route which starts at a given city, visits all the others, and returns to its starting point.

Let c_{ij} be the travel cost between city i and city j, and, for $i < j$, let x_{ij} be 1 if cities i and j are adjacent in the tour, and 0 otherwise. Then the objective function is $\sum_i \sum_{j>i} c_{ij}x_{ij}$ and the following fundamental inequalities hold: $x_{ij} \geq 0$; for all i, $\sum_{j>i} x_{ij} = 2$. Dantzig, Fulkerson and Johnson solved the 49-city instance by solving a sequence of linear programs, starting with the fundamental inequalities, and adding successive cuts as required to exclude fractional optimal solutions. Most of these cuts were of one of the following two forms, where S is a subset of $\{1, 2, \ldots, n\}$:

(a) $\sum_{i,j\subseteq S, i<j} x_{ij} \leq |S| - 1$, where $S \subset \{1, 2, \cdots, n\}$

(b) $\sum_{i<j, |\{i,j\}\cap S=1} x_{ij} \geq 2$,

but a few additional cuts of an *ad hoc* nature were required in order to exclude fractional optimal solutions.

This breakthrough by Dantzig, Fulkerson and Johnson stimulated great interest in the use of cutting-plane methods for integer programming problems. Today cutting-plane methods have become a reliable tool for solving general zero-one integer programming instances, as well as a variety of combinatorial problems with special structure.

Over the years the art of solving large traveling-salesman problems by cutting-plane methods (sometimes combined with partial enumeration of cases) has risen to a very high degree of sophistication. Instances with a few thousand cities can often be solved to optimality within minutes or hours. As of 2006 the largest instance with a proven optimal solution (excluding specially constructed ones with a trivial solution) had 24,978 cities. Applegate, Bixby, Chvatal and Cook [9] report the construction of a tour that is provably within 0.112% of optimal for an instance with 1,904,711 cities, at a cost of 256.1 CPU days of computer time.

5.3. Combinatorial problems as linear programs

In principle any zero-one integer program can also be written as a linear program by adjoining a finite number of cuts to the original linear inequalities. In some notable cases there is an elegant interpretation of the minimal set of required cuts, and the discovery of these cuts illuminates the combinatorial structure of the problem at hand. Such a discovery occurs in the seminal paper "Paths, Trees and Flowers" [23] by Jack Edmonds, which concerns the construction of maximum matchings. Additional examples of this approach are given in [24], in several other papers by Edmonds, and in the books [37,39,40].

A matching in a graph is a set of edges, no two of which meet at a common vertex. In the cardinality matching problem the goal is to construct a matching with a maximum number of edges. In the weighted matching problem each edge has a positive weight, and the goal is to find a matching of maximum total weight.

A graph is called *bipartite* if its vertex set can be partitioned into two parts, such that each edge has an end point in each part; equivalently, a graph is bipartite if all its cycles are of even length. In the special case of a bipartite graph the weighted matching problem is equivalent to the assignment problem, and can therefore be solved as a linear program in accordance with the aforementioned result in [15].

In [23] Edmonds gave a polynomial-time algorithm for the cardinality matching problem in a general graph. His result can be derived either by combinatorial reasoning or through the lens of linear programming. The starting point for the combinatorial approach is the following observation. Let M be a matching in the graph G, and let $\bar{M}$ be the set of edges of G not in M. If M is not of maximum cardinality then there is an *augmenting path* relative to M: *i.e.*, a simple path which begins and ends at a vertex that is not covered by M, and in which the edges of $\bar{M}$ and M alternate.

Given a matching M and an augmenting path P, the set of edges $M \oplus P$, where $\oplus$ denotes "exclusive or", is a matching, and $|M \oplus P| = |M| + 1$. Thus, one can construct a maximum-cardinality matching by augmenting

along augmenting paths as long as possible; however, in the non-bipartite case, it is tricky to find an augmenting path. Edmonds gave a subtle efficient algorithm for finding an augmenting path, based on repeatedly discovering a certain type of maximally matched odd-cardinality set of vertices called a blossom, contracting the blossom to a single vertex, and searching for an augmenting path in the resulting graph. This yielded a polynomial-time algorithm for the maximum-cardinality matching problem.

To extend this result from the cardinality matching problem to the weighted matching problem an approach based on linear programming is useful. One can write the weighted matching problem as a zero-one integer program with a variable x_e for each edge. The objective function to be minimized is $\sum c_e x_e$ where c_e is the weight of edge e, and there is a linear constraint for each vertex v, stating that the sum of the x_e on edges incident with v is at most 1. However, the extreme points of this linear program are not necessarily integral. The key observation is that we can get a linear program with integral extreme points by adding the *odd-set constraints*, which state that, if S is a set of of $2r + 1$ vertices for some positive integer r, then the sum of the x_e over all edges with both end points in S is at most r. Once these additional odd-set constraints are added the system of linear inequalities turns out to have integer extreme points, so that the weighted matching problem can be approached as a linear program, and a polynomial-time combinatorial algorithm for the weighted matching problem can be derived by emulating the primal–dual algorithm, a variant of the simplex method developed by Dantzig, Ford and Fulkerson [20].

5.4. Dantzig's views on integer programming

According to my colleague Ilan Adler, who was one of Dantzig's first Ph.D. students at Stanford, George emphasized in his classes that mixed integer programming could be regarded as essentially a universal format for expressing combinatorial optimization problems. In his paper "On the Significance of Solving Linear Programs with Some Integer Variables" [17] he gave a number of techniques for expressing various types of complex constraints as systems of linear inequalities in which some variables are constrained to be integers. The constraints include dichotomies, k-fold alternatives, selection from many pairs of regions, discrete variable problems, conditional constraints and finding a global minimum of a concave function. In particular, he showed that the fixed charge problem, the traveling salesman problem, the orthogonal Latin square problem and the problem of four-coloring a map could be expressed as mixed integer programs.

Although George's discussion of the universality of mixed integer programming was informal, it can be viewed as a precursor of the work on *NP*-completeness, which emerged in the early 1970s [13,32,25]. Let K and L be subsets of $\{0, 1\}^*$. K is called *polynomial-time transformable* to L if there is a polynomial-time function $f : \{0, 1\}^* \to \{0, 1\}^*$ such that, for all $x \in \{0, 1\}^*$, $x \in K$ if and only if $f(x) \in L$. A set L in *NP* is called *NP-complete* if every problem in *NP* is transformable to L.Thus the *NP*-complete problems are universal; they have enough generality to encompass every problem in *NP*. If L is *NP*-complete then L lies in P if and only if $P = NP$.

Given a combinatorial optimization problem we can usually describe a closely related feasibility problem of essentially the same complexity. If we can show that this feasibility problem is *NP*-complete, it follows that the optimization problem is solvable in polynomial time if and only if $P = NP$. Using this approach we can show that a large class of combinatorial optimization problems, including zero-one integer programming and the traveling-salesman problem, are solvable in polynomial time if and only if $P = NP$. In practice, this is taken as evidence that no general method of solving such a problem can avoid combinatorial explosions; however, the theoretical question of whether $P = NP$ remains open.

To close on a personal note, I would like to acknowledge that Dantzig's views on the universality of integer programming were among my motivations for exploring NP-completeness.

Acknowledgement

Many thanks to Ilan Adler for general advice and in particular for recreating for me his experiences as George Dantzig's Ph.D. student at Stanford.

References

[1] I. Adler, The expected number of pivots needed to solve parametric linear programs and the efficiency of the self-dual simplex method, Technical Report, University of California at Berkeley, May 1983.

[2] I. Adler, G.B. Dantzig, Maximum diameter of abstract polytopes, Mathematical Programming Study 1 (1974) 20–40.
[3] I. Adler, G.B. Dantzig, K. Murty, Existence of A-avoiding paths in abstract polytopes, Mathematical Programming Study 1 (1974) 41–42.
[4] I. Adler, V. Koltun, On the relation between arrangement polytopes and the simplex method, Working paper, 2007.
[5] I. Adler, R.M. Karp, R. Shamir, A simplex variant solving an mXd linear program in $O(\min(M^2, d^2))$ expected number of pivot steps, Journal of Complexity 3 (1987) 372–387.
[6] I. Adler, N. Megiddo, A simplex algorithm whose average number of steps is bounded between two quadratic functions of the smaller dimension, Journal of the Association for Computing Machinery 32 (1985) 871–895.
[7] M. Agrawal, N. Kayal, N. Saxena, PRIMES is in P, Annals of Mathematics 160 (2) (2004) 781–793.
[8] N. Amenta, G. Ziegler, Deformed products and maximal shadows of polytopes, in: B. Chazelle, J.E. Goodman, R. Pollack (Eds.), Advances in Discrete and Combinatorial Geometry, in: Contemporary Mathematics, vol. 223, Amer. Math. Soc., 1999, pp. 57–90.
[9] D. Applegate, E. Bixby, V. Chvátal, W.C. Cook, Implementing the Dantzig–Fulkerson–Johnson algorithm for large traveling-salesman problems, Mathematical Programming 97 (2003) 91–153.
[10] D. Bertsimas, J.N. Tsitsiklis, Introduction to Linear Optimization, Athena Scientific, Belmont, MA, 1997.
[11] K.H. Borgwardt, Untersuchungen zur Asymptotik der mittleren Schrittzahl von Simnplexverfahren in der linearen Optimierung, Ph.D. Thesis, Universität Kaiserslautern, 1977.
[12] K.H. Borgwardt, The Simplex Method: A Probabilistic Analysis, in: Algorithms and Combinatorics, vol. 1, Springer-Verlag, 1980.
[13] S.A. Cook, The complexity of theorem proving procedures, in: Proc. 3rd ACM Symp. on the Theory of Computing, 1971, pp. 151–158.
[14] R.W. Cottle, The Basic George B. Dantzig, Stanford Business Books, Stanford, CA, 2003.
[15] G.B. Dantzig, Application of the simplex method to a transportation problem, in: T.C. Koopmans (Ed.), Activity Analysis of Production and Allocation, 1951, pp. 339–347.
[16] G.B. Dantzig, On the shortest route through a network, Management Science 6 (1960) 187–190.
[17] G.B. Dantzig, On the significance of solving linear programming problems with some integer variables, Econometrica 28 (1960) 30–44.
[18] G.B. Dantzig, Linear Programming and Extensions, Princeton University Press, Princeton, NJ, 1963, Revised edition 1966; fourth printing 1968.
[19] G.B. Dantzig, All shortest routes in a graph, in: P. Rosenstiehl (Ed.), Theorie des Graphes, Dunod, Paris, 1966, pp. 91–92.
[20] G.B. Dantzig, L.R. Ford, D.R. Fulkerson, A primal–dual algorithm for linear programs, in: H.W. Kuhn, A.W. Tucker (Eds.), Linear Inequalities and Related Systems, in: Annals of Mathematics Study, vol. 38, Princeton University Press, Princeton, NJ, 1956, pp. 171–181.
[21] G.B. Dantzig, D.R. Fulkerson, On the max-flow min-cut theorem of networks, in: H.W. Kuhn, A.W. Tucker (Eds.), Linear Inequalities and Related Systems, in: Annals of Mathematics Study, vol. 38, Princeton University Press, Princeton, NJ, 1956, pp. 215–221.
[22] G.B. Dantzig, D.R. Fulkerson, S.M. Johnson, Solution for a large-scale traveling-salesman problem, Journal of the Operations Research Society of America 2 (1954) 393–410.
[23] J. Edmonds, Paths, trees and flowers, Canadian Journal of Mathematics 17 (1965) 449–467.
[24] J. Edmonds, Matroids, submodular functions and certain polyhedra, in: R.K. Guy, H. Hanai, N. Sauer, J. Sch"onheim (Eds.), Combinatorial Structures and their Applications, Gordon and Breach, New York, 1970, pp. 69–87.
[25] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of *NP*-Completeness, W.H. Freeman, San Francisco, 1979.
[26] R. Gomory, Essentials of an algorithm for integer solutions to linear programs, Bulletin of the American Mathematical Society 64 (5) (1958).
[27] M. Haimovich, The simplex method is very good!: On the expected number of pivot steps and related properties of random linear programs, Technical Report, Columbia University, April 1983.
[28] E. Hazan, N. Megiddo, The arrangement method for linear programming is equivalent to the phase-one method, 24, February 2007 (manuscript).
[29] G. Kalai, A subexponential randomized simplex algorithm, in: Proc. 24th Ann. ACM Symp. on Theory of Computing, May 1992, pp. 475–482.
[30] G. Kalai, D. Kleitman, A quasi-polynomial bound for the diameter of graphs of polyhedra, Bulletin of the American Mathematical Society 26 (1992) 315–316.
[31] N. Karmarkar, A new polynomial time algorithm for linear programming, Combinatorica 4 (1984) 373–395.
[32] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller, J.W. Thatcher (Eds.), Complexity of Computer Computations, Plenum Press, New York, 1972, pp. 85–103.
[33] J.A. Kelner, D.A. Spielman, A randomized polynomial-time simplex algorithm for linear programming, in: Proc. 38th Annual ACM Symposium on Theory of Computing, 2006, pp. 51–60.
[34] L.G. Khachiyan, A polynomial algorithm in linear programming, Doklady Akademia Nauk SSSR (1979) 1093–1096.
[35] V. Klee, G.J. Minty, How good is the simplex algorithm? in: O. Shisha (Ed.), Inequalities - III, Academic Press, 1972, pp. 159–175.
[36] V. Koltun, The arrangement method for linear programming, Computer Science Department, Stanford University, November 2005 (manuscript).
[37] E. Lawler, Combinatorial Optimization: Networks and Matroids, Holt, Rinehart and Winston, New York, 1976, Reprinted by Dover, Mineola, NY, 2001.
[38] N. Megiddo, Improved asymptotic analysis of the average number of steps performed by the self-dual simplex algorithm, Mathematical Programming 35 (2) (1986) 140–172.
[39] C.H. Papadimitriou, K. Steiglitz, Combinatorial Optimization: Algorithms and Complexity, Prentice-Hall, Anglewood Cliffs, NJ, 1982, Reprinted by Dover, Mineola, NY, 1998.
[40] A. Schrijver, Combinatorial Optimization: Polyhedra and Efficiency, Springer-Verlag, Berlin, 2003.
[41] S. Smale, On the average number of steps in the simplex method of linear programming, Mathematical Programming 27 (1983) 241–262.
[42] D.A. Spielman, S.-H. Teng, Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time, Journal of the ACM 51 (2004) 385–463.

[43] E. Tardos, A strongly polynomial minimum cost circulation algorithm, Combinatorica 5 (3) (1985) 247–256.
[44] M.J. Todd, Polynomial expected behavior of a pivoting algorithm for linear complementarity and linear programming problems, Mathematical Programming 35 (1986) 173–192.
[45] V. Vazirani, Approximation Algorithms, Springer-Verlag, Berlin, 2001.