Editor's Note: This article was intended for the 25th Anniversary Issue of the *Journal*, but was unavailable at the time of publication. Because we feel that some readers might enjoy exploring with the authors the profound implications of this classical problem, we are including it here.

Harlow Freitag, Editor

Aspects of the traveling salesman problem

by M. Held
A. J. Hoffman
E. L. Johnson
P. Wolfe

For fifty years the traveling salesman problem has fascinated mathematicians, computer scientists, and laymen. It is easily stated, but hard to solve; it has become the prototypical hard problem in theoretical computer science. A large part of the extensive research conducted by IBM in the broad area of optimization, or mathematical programming, contributed to or was inspired by aspects of this challenging problem. This article reviews some of that work as well as recent developments in techniques that were used on the largest traveling salesman problem ever solved.

1. Introduction

The first statement of the Traveling Salesman Problem (TSP) we know of was made in 1930 by the Viennese mathematician Karl Menger. It arose in connection with "A new definition of curve length" that Menger proposed: that the length of a curve be defined as the least upper bound of the set of all numbers that could be obtained by taking each finite set of points of the curve and determining the length of the shortest polygonal graph joining all the points. "We call this the messenger problem, because in practice the problem has to be solved by every postman, and also by many travelers: finding the shortest path joining all of a finite set of points whose distances from each other are given. Of course, the problem can be solved by a finite number of trials. No rule is known that would reduce the number of trials to less than the number of permutations of the given points. The rule of proceeding from the origin to the nearest point, then to the nearest point to that, and so on, does not generally give the shortest path" [1].

The new definition of curve length may not have panned out, for it does not appear in Menger's later work. While its computational tractability, or lack thereof, has little bearing on whether it suited his mathematical needs for a definition of curve length, he certainly saw the practical difficulty of solving a problem of reasonable size. Given n points, and the distances c_{ij} between each pair of points i, j, one path would be determined by a particular ordering $i_1, i_2, \cdots, i_n$ of the points, and the length of the path would be $c_{i_1 i_2} + c_{i_1 i_3} + \cdots + c_{i_{n-1} i_n}$. In a "brute force" approach, that calculation would be made for each ordering of the points (an insignificant half of the orderings could be skipped in the *symmetric* case that $c_{ij} = c_{ji}$ for all i, j, and one went to the great trouble of checking whether an ordering had already been checked in the reverse direction). The number of such orderings is $n!$, the innocent-looking function whose growth with n is explosive: $5! = 120$; $10! = 3\,628\,800$, and $25! = 16 \times 10^{25}$ (greater than the age of the universe in microseconds). As Menger indicates, these problems cannot be solved unless rules can be devised that give numbers of operations far smaller than these.

The next mention we know of the problem is in 1934, followed by perhaps its first statement in connection with a practical problem: Merrill Flood, in his 1956 review article [2], writes: "This problem was posed, in 1934, by Hassler

Whitney in a seminar talk at Princeton University. There are as yet no acceptable computational methods...." Mentioning related problems from the theory of graphs, Flood continues: "I am indebted to A. W. Tucker for calling these connections to my attention, in 1937, when I was struggling with the problem in connection with a schoolbus routing study in New Jersey."

There is a small difference between Menger's statement and Flood's, which is now the customary statement: Flood seeks a closed polygonal path, or *tour*—the salesman visits all the cities and returns to his point of origin. The cost to be minimized is thus $c_{i_1 i_2} + \cdots + c_{i_{n-1} i_n} + c_{i_n i_1}$. A *permutation* P is a function from the integers $1, \cdots, n$ onto the same integers [i.e., $P(i) \neq P(j)$ if $i \neq j$]; and it represents a tour if it is a *cyclic* permutation, i.e., is such that n successive applications of P to any of those integers generates all of them. We can then state the TSP as the problem of minimizing $\Sigma_i c_{iP(i)}$ over all cyclic permutations P. (We do not assume symmetry, i.e., $c_{ij} = c_{ji}$.) While this holds for Euclidean distances, it fails to hold in many applications. As a simple example, c_{ij} might represent the travel time from i to j in hilly country. The closed path represented by a permutation is called a *directed* tour. When the matrix (c_{ij}) is symmetric, only symmetric permutations need be considered, and one speaks of *undirected* tours.

Partly because of its simple statement and its quickly acquired reputation for difficulty, the problem has fascinated a whole generation of professionals and laymen; the former have published some 300 technical papers on aspects of it. Much of that fascination for professionals is due to the fact that other combinatorial problems, sounding much like the TSP, have yielded to analysis, and practical algorithms for solving them, even in large-scale cases, *have* been devised. (The "assignment problem" of the next section is a fine example.) By the early sixties many of the ideas that have been used on the problem had been proposed; they are reviewed in Ralph Gomory's paper [3]. Nevertheless the TSP remained difficult and has become, indeed, the prototype of a "hard" combinatorial problem, the most cited example of an NP-complete problem in computational complexity (see the next section); and perhaps it has inspired much of the work of that field. Likewise, all the work that has been done on it has had important consequences for the whole area of optimization. It was attacked by procedures which found much more general use in the broad field of integer programming and even in nonlinear programming. It seems that most of the research on optimization methods done at IBM is connected with this problem, and a review of the history of efforts to solve and understand it covers a large part of IBM researchers' work in optimization.

This account focuses on IBM contributions to mathematical and computational problems related to the TSP, with mention of some work from other sources that was truly seminal. For reasons of space we have had to omit any description of the many applications, some of them not at all obvious, to which the TSP has lent itself, as well as a great deal of IBM work in the broader field of mathematical techniques of optimization; but enough remains to show that a great deal of progress has been made in coping with "hard" problems and that good, hard problems can inspire good research.

2. Complexity and heuristics

During the last ten years the notion of *NP-completeness* has swept through computer science and allied fields. We do not detail that complex subject here. For a complete treatment the reader is referred to the first two chapters of the book by Garey and Johnson [4]. We mention only that the TSP, which is the first problem described in their book, is a member of the class of what are called NP-complete problems. These are problems for which there is no known algorithm which is "good" (in the sense of being "polynomially bounded": having a running time bounded by a polynomial in the length of the input data string needed to define any particular instance of the problem). Further, if a "good" algorithm existed for any of them, a good algorithm would exist for each of them. These problems come from a variety of important fields, including database design, network design, compiler implementation, and operations research. While the question of finding such an algorithm is still open, most experts believe it will never happen.

Building on a foundation laid by Stephen Cook, Richard Karp established the polynomial equivalence of many combinatorial problems with the TSP. His basic paper [5], presented at the first IBM symposium on computational complexity in 1971, described 21 such problems. We like to think that much of his interest in the area was spurred by his earlier work at the IBM Thomas J. Watson Research Center on the TSP itself [6, 7].

The NP-complete results have given a new impetus to *heuristic* methods for solving these problems, which can fairly be viewed as "intractable" in the sense that there seems to be little use, at present, in seeking algorithms for them which yield assured solutions but whose laboriousness does not grow exponentially with problem size. We may define *heuristic* as an algorithm for which we know no mechanism inherent to the algorithm for establishing the optimality of the results it produces. Much work has been done on heuristic algorithms for the TSP and related problems, and some of these have been remarkably effective, as was later determined when methods arose with which optimality could be established. We do not review these here (since most IBM work has focused on the more demanding task of guaranteeing solutions) except to mention the important work of Lin and Kernighan [8]. Their heuristic produces excellent near-solutions to the TSP which were used to provide starting points for the very-large-scale problems of Section 11.

The likelihood that heuristic methods may be the only feasible approach to many combinatorial problems of large size has certainly made those methods more interesting and respectable, and has given impetus to an entire new field of research: obtaining performance guarantees for heuristic algorithms, both in worst-case and probabilistic senses. Here too the TSP has played a seminal role. Karp's 1977 paper [9] established the general definitional framework for the probabilistic analysis of heuristic algorithms, and applied it to the TSP.

3. Linear programming

The general linear programming (LP) problem plays a central role in many of the computational methods that have been used on the TSP. It may be stated: Minimize the linear function $z = cx$ subject to the constraints $Ax = b$, $x \geq 0$, or

$$\min \{cx : Ax = b,\ x \geq 0\}, \tag{1}$$

where x is a variable n-vector and the given data c, A, b are, respectively, an n-vector, an m-by-n matrix, and an m-vector. (We juxtapose c and x to denote an inner product, and A and x to denote matrix-vector multiplication. We note that standard devices [10] convert problems having inequalities $Ax \leq b$, or variables x not constrained as here, into this form.)

The term "programming" does not refer to the writing of programs for a computer, but to *planning*. About 1944 a group of scientists including the mathematician George Dantzig was brought together by the U.S. Air Force under the project title "Scientific Computation of Optimum Programs," with the mission of applying mathematical techniques to large-scale planning ("programming") problems. They found the simple (in principle) LP statement to be an adequate model for a vast field of applications, and thus called the procedure *linear programming*. In 1947 Dantzig invented the algorithm he called the "simplex method" for solving the models numerically, and that algorithm, greatly improved, and redeveloped for each new generation of computer, is the one in use today.

As one of the pioneers has said, "mathematical programming and computing have been contemporary in an almost uniquely exact sense. Their histories parallel each other year by year in a remarkable way" [11]. The first general-purpose linear programming routine was written for the SEAC in 1951. On Dantzig's moving to the RAND Corporation in Santa Monica, California, in 1952, he, W. Orchard-Hays, and L. Cutler began development of the first large-scale software for linear programming on RAND's IBM 701 computer. On introduction of the IBM 704 in 1955, the RAND group, with support from IBM in machine time and personnel, developed a routine for this machine which, widely distributed by the SHARE organization, set the formats and standards for LP software from that point on. It could handle problems having up to 250 linear relations, a staggering number for the time, and found immediate and widespread use, first in the oil industry. Refinery operation, storage, and distribution problems could all be modeled as LP problems, and solving them turned out to be highly profitable. We note that an IBM Petroleum Conference held in Endicott, New York, in October 1953 was host to a number of papers on LP applications.

Development of algorithms and software for LP has never ceased. Every large computer manufacturer has provided routines for LP and some of the extensions we discuss. IBM has built its own LP software since the early sixties; the current version, MPSX [12], can handle more than 16 000 equations and an almost unlimited number of variables. It and earlier IBM LP software are heavily used in thousands of installations.

An important mathematical aspect of LP is *duality*. It was found (see, e.g., [10]) that *Lagrange multipliers* (the m-vector y below) could be used in LP by formulating the minimax problem

$$\min_{x \geq 0} \max_{y} [cx - (Ax - b)y], \tag{2}$$

easily seen to be equivalent to (1). A fundamental theorem asserts the equality of (2) to

$$\max_{y} \min_{x \geq 0} [by - (A^T y - c)x], \tag{3}$$

which is equivalent to the new LP problem

$$\max \{by : A^T y \leq c\}. \tag{4}$$

The problem (4) is called the *dual* of the original LP problem, and the equivalence of the two LP problems is stated in the *duality theorem of linear programming*: If either the problem (1) (called in this context the *primal* problem) or the dual problem has a solution, then so does the other; and then the values of cx and by are equal.

A solution of the dual problem, along with that of the primal, is automatically produced when the primal problem is solved by modern LP software. That fact provides an ironclad test of optimality: any x and y satisfying the constraints of the primal and dual, respectively, must satisfy the inequalities $cx \geq (A^T y)x = by$, so when such x and y are found with $cx = by$, the required minimum and maximum are known to be at hand; checking optimality is trivial.

Another important theoretical feature of LP is the fact that if a problem has any solution at all, then it has one which is an *extreme point* of the convex polyhedron defined by the constraints of (1), that is, a solution which is not a convex combination of any other two points satisfying the constraints. Further, the simplex method always yields such an extreme point solution, a fact we use later.

4. Permutation problems

LP provides a powerful tool for problems involving permutations. We can represent a permutation P as a matrix of real numbers as follows: the n-by-n real matrix X will have the entry $x_{ij} = 1$, where $P(i) = j$, and zero elsewhere.

The matrix X thus has exactly one 1 in each row and the same in each column, and any matrix consisting entirely of zeros and ones (0–1, for short) having that property is square, and represents a permutation. We call such matrices *permutation* matrices. Another expression of this property is

$$\sum_j x_{ij} = 1 \quad \text{for all } i,$$

$$\sum_i x_{ij} = 1 \quad \text{for all } j,$$

$$x_{ij} \geq 0 \quad \text{for all } i, j. \tag{5}$$

Noting that

$$\sum_i c_{iP(i)} = \sum_{ij} c_{ij} x_{ij} \tag{6}$$

when X represents the permutation P, we could state a minimization problem over the set of *all* permutations as that of minimizing the right-hand side of (6) for all 0–1 matrices satisfying the constraints (5). Since the requirement $X \geq 0$ evidently holds, we have formulated it as an LP problem, save for the requirement that only zeros and ones be used, rather than any nonnegative numbers satisfying the constraints.

For such a problem, we can take advantage of the fact that our LP software will find an extreme point solution of the problem. A celebrated theorem of Garrett Birkhoff states that the extreme points of the polyhedron defined by (5) are precisely all the permutation matrices. Thus our solution of the LP problem will automatically be 0–1, and the required permutation is found. (Other problems whose LP solutions are also automatically 0–1 are mentioned in Section 10.)

The problem just stated is a useful one, commonly known as the "assignment" problem. Suppose that any of n people can do any of n different jobs (but not all equally well), and let c_{ij} be the net cost of assigning person i to job j. The problem is that of finding an assignment, or permutation, P so that the total cost $\Sigma_i c_{iP(i)}$ is minimized. Large assignment problems—$n > 8000$—could thus be solved by present LP software, although the problem has such special features that certain refinements of standard LP methods operate with even much greater efficiency.

One might hope to generalize the approach above: Given any subset of the set of all permutations, find (which is possible in principle) a system of linear equations and inequalities which defines the convex hull of that set, and solve the resulting LP problem (again possible in principle); the answer is the permutation sought. Unhappily, in most cases we do not know how to do this explicitly. The only difference between the assignment problem and the TSP is that any permutation may be an assignment, while only cyclic permutations can be tours; but we do not know how to write down all the linear relations that define the convex hull of the set of all tours. (It has been done for small problems. The number of such constraints is vast, and is known to grow exponentially.) However, all is not lost: It may be possible to find enough of those constraints to define that part of the convex set lying close to the optimum solution, and thus solve the problem without finding all the constraints. Such a procedure has succeeded, and is discussed in Section 5.

A fruitful offshoot of these considerations has been a wide variety of mathematical results linked to LP representations of combinatorial problems, which are reviewed in Section 10.

Alternatively, if it were possible to ensure that X would remain 0–1, one could write a set of linear constraints which exclude unsuitable permutations for the TSP. Let S denote any proper subset of the points $1, \cdots, n$. The requirement

$$\sum_{i,j \in S, i \neq j} x_{ij} \leq |S| - 1, \tag{7}$$

where $|S|$ denotes the cardinality of S, is called a "subtour elimination" constraint, and ensures that the permutation does not have a cycle on S, that is, a closed path running just through the points of S, since the left-hand side is just the total number of single segments joining points of S. Thus if all constraints of the form (7) can be enforced (we may, incidentally, omit subsets S for which $|S| > n/2$), together with (5), and X can be made 0–1, the linear formulation will suffice.

Ignoring for the moment the extraordinary number—2^{n-1}—of these constraints (there are other, more complicated, formulations which make their number more nearly reasonable)—the question is, can the variables be forced to be 0–1? Of course, the answer is "yes," for otherwise this paper would not have been written; most of the rest of it deals with just that question.

5. The 42-city problem

The first TSP of serious size was solved in 1954 by George Dantzig, Ray Fulkerson, and Selmer Johnson of the RAND Corporation [13]. The method used, necessary at the time, was a hybrid: repetitive use of LP as we describe here and a certain amount of direct human manipulation of intermediate results. They chose a problem with symmetric distances (road distances between 42 major cities in the United States), which modestly reduces the computational demands: Instead of dealing with full permutation matrices and square numerical matrices with n^2 entries, only $n(n-1)/2$ entries are needed. Data x_{ij} are recorded only for $i < j$. A one in position i, j for $i < j$ signifies a path segment joining cities i and j, the direction of traversal being ignored. The principles they used are just the same for the unsymmetric problem.

The most important tool they developed was the *cutting plane*. While that term was not coined at RAND, but later

by Ralph Gomory at Princeton (see below), the concept became an essential ingredient in the application of LP to integer problems. Its basic principle is as follows:

> If the solution of an approximation (say, R) to an optimization problem (say, Q) does not satisfy all the constraints of Q, adjoin to R a linear constraint which the solution of R does not satisfy but which is satisfied by all candidates for the solution of Q.

By the theorem of separating hyperplanes, this can always be done if the constraints of R define a convex set; the adjoined linear constraint defines the cutting plane, which cuts away a portion of R not agreeing with the requirements of Q. The RAND researchers began with an LP (a problem R) for the TSP using only the constraints $X \geq 0$ and the "degree two" constraints

$$\sum_{i<j} x_{ij} + \sum_{j<i} x_{ij} = 2, \qquad i = 1, \cdots, n, \tag{8}$$

together with an heuristically generated tour which, of course, provided a basic feasible solution to the LP. Linear programming pivot steps were then performed until either an optimal tour was obtained or until a pivot step would lead to a vertex which was not a tour (owing to the occurrence of subtours) but which was still 0–1. In the latter case the LP was augmented by the appropriate subtour elmination constraint (7), forming a new R. This process continued until a pivot step would lead to a solution which was not 0–1. At this point, R was solved to optimality, providing a lower bound for the optimal value of the TSP. A *branching* strategy was then used: A promising variable x_{ij} whose value lay strictly between 0 and 1 was chosen, and two separate cases studied, one in which the variable was set to 0 and another in which it was set to 1. Frequently an *ad hoc* argument based on inspection of a near-tour could be used to eliminate one of these cases; if not, both could be pursued. Finally, a simple argument based on the reduced costs of the LP solution to R frequently could be used to show that one of the two settings of the variable would necessarily result in a solution whose cost was greater than that of the best tour so far found, so that that setting could be eliminated.

Their *ad hoc* argument presaged the kind of method later formalized as "enumeration" in integer programming, and the argument based on costs was a forerunner of the general "branch and bound" procedure, both discussed in Section 8; but it was the idea of the cutting plane that first became important for very wide classes of problems.

Commenting on the RAND work on the TSP in 1964, Gomory wrote: "I do not see why this particular approach stopped where it did. It should be possible to use the same approach today, but in an algorithmic manner. We no longer have to be artistic about generating the separating hyperplanes or cuts, since this is now done automatically in integer programming . . ." [3]. He was right; the whole procedure could be effectively automated. In the next section we describe how this was first done for general integer programming problems. However, it took fifteen more years before the insights that were being developed could make full automation of a large-scale TSP solution a reality.

6. Cutting planes

The early work at the RAND Corporation on the TSP reviewed in the previous section showed the utility of LP for these problems and the value of skillfully constructed cutting planes. At that time, however, there was no systematic method for generating cutting planes that would cut off noninteger solutions of the approximating LP problem. That was first provided by Gomory [14–16], who gave a method with proven finite convergence for general all-integer problems.

An integer programming (IP) problem is the LP problem (1), with the additional restriction that x_j be an integer for $j \in J$, where J is a subset of $\{1, 2, \cdots, n\}$. In a *pure* integer program J consists of all of $\{1, 2, \cdots, n\}$, while for a *mixed* integer program J is a proper subset of them.

IP problems are almost always more difficult than LP problems, for they incorporate discrete and nonconvex aspects of problems which cannot be modeled by LP; Gomory's review paper [17] covers many aspects of that. Certain important LP problems, describing flows in networks, have the total unimodularity property (see Section 10) and thus automatically give integer optimum solutions; but one does not expect a typical IP problem to fall in that class, and in general much work is required to get from an LP solution (with fractional values) to an integer solution (with all x_j integer, $j \in J$).

The *Gomory cut* is best described in terms of the "standard" or "tableau" form of the simplex method (see, e.g., [10]). Briefly, at each iteration a subset of m variables x_j: $j \in B = \{j_1, \cdots, j_m\}$ is designated *basic* and the remaining set, say N, *nonbasic*. The m equations

$$x_{j_i} + \sum_{j \in N} \bar{a}_{ij} x_j = \bar{b}_i \geq 0, \qquad i = 1, \cdots, m, \tag{9}$$

equivalent to the starting system of equations, permit the expression of the basic variables explicitly in terms of the nonbasic, and yield the *basic feasible solution* $\{x_{j_i} = \bar{b}_i: i = 1, \cdots, m\}$ of the problem on setting $x_j = 0$, j $\in N$. If all $\bar{b}_i$ are integral, then so is the basic solution; otherwise, choosing some nonintegral $\bar{b}_1$, the cutting plane

$$\sum_{j \in N} f(\bar{a}_{1j}) x_j \geq f(\bar{b}_1) \tag{10}$$

is adjoined to the system, where $f(c)$ denotes the fractional part of c: $f(c) = c - p$, where p is the largest integer $p \leq c$.

It is easy to check that any integer solution of (9) also solves (10), while the current basic feasible solution does not, so that (10) truly gives a cutting plane. Gomory was able to

show that repeated adjunction of these cuts, together with a suitable organization of the dual simplex method calculation, would terminate with an integer solution after a finite number of steps.

Unhappily, pure cutting plane methods have not been adequately developed for mixed integer problems; and they display a certain lack of stability on even small pure problems. While some are solved quickly, similar problems may just run on and on. This behavior is not well understood, but then it is not surprising since the good behavior of the simplex method on linear programs is not well understood either; and what cutting plane methods do is convert the problem to a linear program with many constraints not specifically identified in advance. Nevertheless, cutting planes continue to play an essential part in the efficient solution of very large integer problems, including the largest TSP ever solved (see Section 11).

In a different direction, cutting planes have been important in solving nonlinear programming problems. A general statement of the problem is

$$\min f_0(x): f_i(x) \leq 0, \qquad i = 1, \cdots, m, \tag{11}$$

where x is the n-vector $(x_1, \cdots, x_n)$. (Of course, if all the functions of (11) are linear, this is just an LP problem.) Denoting the gradient of the function f_i at any point x by $\dot{f}_i(x)$, any nonlinear function of (11) can be approximated by the family of linear functions

$$f_i(x^k) + \dot{f}_i(x^k)(x - x^k), \qquad i = 1, \cdots, K$$

if the points $x_1, \cdots, x^K$ are suitably chosen. The solution of the resulting LP problem, if not a sufficiently good approximation to a solution of (11), can be adjoined to the existing collection as x^{K+1}, and the procedure repeated. The rate of convergence of this kind of procedure, first proposed in 1957, was established for convex functions by Philip Wolfe [18], who refers to the extensive literature on this approach.

The cutting-plane methods all add constraints to an approximate problem R to make it more nearly like the given problem Q; that is, they add rows to the matrix A defining the linear problem. In view of LP duality, that procedure is equivalent to adding columns to the transpose of A which defines the dual problem. Such a general scheme, devised from other considerations by Dantzig and Wolfe [19] for the solution of very large LP problems of a certain "decomposable" structure, is called *column generation*, and has found hosts of applications in large-scale LP and NLP. One of these was the work of Gomory and Hu [20] on communication network design, and another the work of Gilmore and Gomory [21] (which was awarded the Lanchester Prize of the Operations Research Society of America for 1963) on the "cutting stock" problem, which also led to some of the further developments sketched in the next section.

7. Knapsacks and facets

A pure integer programming (IP) problem in nonnegative variables having only one constraint (A has but a single row), $\Sigma a_j x_j \leq a$, is called a *knapsack* problem. Despite its simple appearance, the knapsack problem is NP-complete; indeed, there are ways to state any pure IP problem as a knapsack problem—the number of constraints is no measure of complexity! Among many other appearances, it occurs as a subproblem, which must be solved many times, in the cutting stock algorithm of Gilmore and Gomory [21]. Their paper [22] devoted to the knapsack problem itself presents several algorithms for solving it, one based on a property of the problem which has had many subsequent ramifications: As the right-hand side a varies, there are only, essentially, a finite number of solutions of the problem; further, there is a periodicity in the values of all but one variable for sufficiently large values of a. This observation, along with his earlier work on cutting planes, led Gomory to the discovery of similar periodicities for the general pure integer problem [23] and to the so-called *group* problem for IP [24].

Gomory then focused on the polyhedra of the convex hull of group solutions as an object of study for IP [25]. In a landmark paper [26] he gave numerous properties of this class of polyhedra, including a powerful characterization of the *facets* (defining inequalities) of these polyhedra in terms of subadditive functions of a real variable.

In subsequent work [27, 28] Gomory and Ellis Johnson continued those studies, including for the first time continuous variables. Johnson [29] and Crowder and Johnson [30] extended this work to the general mixed integer group problem, giving some insight into why cutting planes had not been successful for the mixed problem and what type of functions needed to be used to successfully address it. This work has been continued to provide a general theory [31] of facets of mixed integer programs.

At the same time, algorithmic developments featured use of subadditive functions for these cyclic group problems, the general group problem, the pure integer problem [32], certain mixed problems, and the knapsack problem [33].

The moral of this highly compressed tale (which omits most of the literature written on the subject at the Thomas J. Watson Research Center during the last ten years) is that study of the structure of the polytopes determining IP problems was difficult, but rewarding; and it is continuing, with significant benefit to practical solution of these important problems.

Meanwhile, intensive work was going on at IBM and elsewhere on identifying more of the facets of specific combinatorial problems (e.g., [34]). While work in the fifties (reviewed by Gomory [3]) had discovered classes of facets for the TSP going far beyond degree two (8) and subtour elimination (7), it was a new class of facets [35, 36] that greatly aided the successful calculations reported in Section 11.

Let K be an odd integer and let $S_0, S_1, \cdots, S_K$ be proper subsets of the cities such that $\{S_1, \cdots, S_K\}$ are pairwise disjoint but each contains at least one node of S_0 and at least one node not in S_0. Then the constraint

$$\sum_{k=0}^{K} \sum_{i,j \in S_k, i \neq j} x_{ij} \leq |S_0| + \sum_{k=1}^{K} (|S_k| - 1) - \left\langle \frac{1}{2} K \right\rangle, \qquad (12)$$

where $\langle . \rangle$ denotes the next highest integer, constitutes a facet of the TSP [35].

Subtour elimination constraints are a very special case of these: If $K = 1$ and $S_0 = \{i\}$ is a single node, then (12) becomes just (7). It is known that, despite the obviously vast number of constraints of the form (12), they do not define the TSP polyhedron; still other facets exist, but characterizations for them have not been found.

8. Enumeration and branching

Despite the complete impossibility of solving the TSP, or any other combinatorial problem of interesting size, by simply enumerating all feasible solutions, the concept of enumeration is at the heart of many of today's solution procedures. They all, of course, incorporate some means of limiting the amount of enumeration that has to be done, and differ from one another primarily in how they go about that.

In the early sixties Michael Held and Richard Karp [6] found that the so-called *dynamic programming* technique could be applied to permutation problems; it reduces the growth rate of the work required to solve the problem from a number like $n!$ to one more like 2^n. They developed a computer program for solving the TSP which was able to solve problems having as many as 13 cities using only internal memory on a 32K computer. They combined this dynamic programming algorithm with a heuristic scheme which then made possible the solution of problems of as many as 50 cities (without, however, assurance that the true solution had been found). The program was so successful (for its time) that IBM issued a press release about it. Despite this early success, dynamic programming has not proven to be a particularly effective technique for solving hard combinatorial problems, except possibly for knapsack problems [22].

In 1963 Little, Murty, Sweeney, and Karel published a landmark paper [37] which first used the term "branch and bound" (BB). A very simple method was proposed and tested for the TSP. The results showed a great deal of promise: Problems of 40 cities could be solved in a few minutes of IBM 7090 computer time.

A "decade of enumeration" followed: A profusion of papers appeared presenting and testing different enumerative methods for the TSP and a host of other combinatorial problems. Notable among these was the work of Spielberg and his colleagues who, in a succession of papers and computer programs (see, e.g., [38–42] gave very efficient enumerative methods for general IP, 0–1 IP, and various versions of the *plant location* problem which, like the TSP, is NP-complete. A great deal of general-purpose software using BB methods for mixed integer programming was also written, including the IBM Program Product MPSX-MIP/370 [43]. Its developers have documented the extensive investigation into choice of method and solution strategy that preceded its design [44]. It continues to be the state-of-the-art software for the solution of general IP problems.

A typical BB algorithm contains three elements. These are separation (partitioning) into subproblems, selection of subproblems (branching), and bounding. Let us state the optimization problem in the form minimize $z(x)$, $x \in S$, where S denotes the set of all objects to be investigated—in the case of the TSP, the set of all tours. We do not attempt to solve the problem directly over S, but divide it successively into smaller and smaller sets to be searched—*separation*. For the TSP separation is usually done by dividing a given set of tours into two subsets: In one subset all tours *must* pass over the link joining a certain pair of cities, and in the other subset no tour may use that link. For 0–1 problems it is done by fixing a single variable at either 0 or 1. As the enumeration proceeds the subsets decrease in size until it finally becomes possible to solve the subproblems. If this happens before their number becomes huge, the problem will be solved. (The language of graph theory is often used to describe the above procedure. The family of subsets is a *tree*; separation is *branching*; and eliminating a subset from consideration, as below, *pruning*.)

The most important possibility for limiting the number of subsets generated is the discovery of a sufficiently high *lower bound* for the subset under investigation: If it can be shown, by any means, that all subsets of the current set can yield no lower a value for z than some value already obtained elsewhere, then that set need no longer be pursued. In integer programming the bounding is typically done by LP: At a particular step, certain integer variables have been (tentatively) fixed; the integer requirement is dropped on the remaining variables, so that the remaining minimization problem is just an LP problem; and its solution, since its constraints are a relaxation of those of the IP problem, must yield a lower bound on all possible solutions having those variables fixed. When suitable strategies are used for the crucial choice of which subsets to explore first, so that high bounds are obtained soon and large areas of the problem are ruled out for searching, the technique is very powerful.

In 1970 Held and Karp developed a BB algorithm for the TSP by using an easily implemented "subgradient" procedure, described in the next section, to quickly compute very sharp bounds. It was so effective that, as they wrote [7], "It is possible for us to do something which has never been done before—to present in their entirety the search trees for large combinatorial problems of this type." The 42-city problem [13], for example, required the enumeration of only 61 of the 33×10^{49} possible tours in the problem.

At present, software that solves large combinatorial problems is based almost exclusively on BB, which underscores the importance of the bounding methods described in the next section.

9. Bounds and subgradients

One of the most computationally useful ideas to arise in the seventies is that many difficult problems can be viewed as easy problems complicated by additional constraints; it leads to a general scheme for producing bounds for BB algorithms which, under the name *Lagrangian relaxation*, has had widespread use. In the form in which it is presently used it follows from the work of Held and Karp [7], who used it to develop a very successful algorithm for the symmetric TSP.

Let the given problem be stated as

$$\min \{cx : Ax = b,\ x \in S\}, \tag{13}$$

where S is one of the subsets of the BB procedure for which a lower bound is needed. In a 0–1 IP, for example, S is the set of all 0–1 vectors whose trial values have not yet been fixed. We form, as in Section 3, a Lagrangian for the problem, and rewrite it as

$$\min_{x \in S} \max_{y} [cx - (Ax - b)y]. \tag{14}$$

It is immediate that (14) is no less than

$$\max_{y} F(y) = \max_{y} \min_{x \in S} [by - (A^T y - c)x], \tag{15}$$

so that the function $F(y)$ defined in (15) serves as a lower bound of (13) for *any* y. [The values of (13) and (15) are not necessarily equal here, as is the case in LP duality. They would be equal if the set S were convex, but the sets in which we are interested are very nonconvex.]

When S is the set of all 0–1 vectors, the calculation of $F(y)$ for any y is trivial. The application to the TSP is subtler. Held and Karp chose S to be the set of all "1-trees," a 1-tree being a collection of intercity links consisting of a *spanning tree* on the cities 2, 3, $\cdots$, n together with two distinct links to city 1. Every tour is a 1-tree, and a 1-tree is a tour only if each city has just 2 links. The latter requirement is just the degree-two constraint (8), which thus serves as $Ax = b$ above. Calculation of $F(y)$ in this case requires finding a minimal spanning tree, a problem for which very fast algorithms are available.

Since a high lower bound for (13) is important, we want a y that makes $F(y)$ large, although finding its exact maximum may not be worth the trouble. Being the infimum of a family of linear functions, F is concave, but not everywhere differentiable. A "differential calculus" of such functions has been developed (see, e.g., [45]) in which the notion of *subgradient*, an extension of the *gradient* for differentiable functions, is central. In our case, a subgradient of F at y is given by Ax_y, where x is that member of S achieving the minimum of (14). A concave function can be maximized by a surprisingly simple procedure: Choose *step lengths* $t_k > 0$ such that $t_k \to 0$ and $\sum_{k=0}^{\infty} t_k = \infty$, choose the starting point y_0, and perform the recursion $y_{k+1} = y_k + t_k g_k$ for $k = 0, 1, 2, \cdots$, where g_k is a subgradient of F at y_k. The values $F(y_k)$ then converge to the maximum value of F.

Held and Karp invented a version of this scheme for their TSP work [7]. The general procedure is due to the Soviet mathematician N. Z. Shor [46], who further developed it in many subsequent publications (e.g., [47]). While even now the rate of convergence of *subgradient optimization*, as this is called in the Western literature, is not well understood, the speed with which a single step can be taken makes it an excellent algorithm for getting approximate solutions. Crowder, Held, and Wolfe [48] demonstrated its practicality for a wide variety of optimization problems. It had long been thought that differentiability was essential for the effective maximization of a concave function. The success of one method for nondifferentiable functions inspired a variety of more refined procedures (e.g., [49, 50]) as well as the publication of a collection of papers on the new subject of "Nondifferentiable Optimization" [50]; but here we wander from the trail of work closely related to the TSP.

10. Combinatorial offshoots

The connection between problems involving permutations and linear programming described in Section 4 has inspired a whole literature connecting LP, especially duality theory, with extremal combinatorial problems of a more general nature.

To describe the general setting, consider the LP problem (1) and its dual (4), and assume that its data A, b, c are all integral. Suppose it is true that the set of all pairs (x, y) which are respectively feasible for primal and dual have a combinatorial meaning if they are integral. If one can prove there exist optimal (x, y) which are integral, then the duality theorem of linear programming proves a combinatorial theorem.

The first explicit instance of this paradigm may have been a proof of the König-Egervary theorem that the largest cardinality of a set S of 1s contained in a (0, 1) matrix M such that no two elements of S are in the same row or column is the smallest cardinality of a set T of rows and columns containing all 1s in M. The first cardinality sought is given by the solution of the LP problem

$$\text{maximize } \sum_{ij} M_{ij} x_{ij} : x_{ij} \geq 0,\ \sum_{j} x_{ij} \leq 1,\ \sum_{i} x_{ij} \leq 1, \tag{16}$$

whose constraints relax those (5) defining permutation problems. If the optimum vertex $X = (x_{ij})$ is integral, then each row and column of X consists either entirely of zeros or of a single one, the remaining entries being zero, and the X sought has as many ones as possible. If (16) is taken as a dual problem, the primal problem is

$$\text{minimize } \sum_{i} u_i + \sum_{j} v_j : u_i \geq 0,\ v_j \geq 0,\ u_i + v_j \geq M_{ij} \quad \text{for all } i, j, \tag{17}$$

which evidently yields the second cardinality sought.

The integrality of optimum primal and dual vectors follows from the fact that the coefficient matrix A governing the inequalities is totally unimodular; i.e., all nonsingular square submatrices have determinant ± 1, which implies that the use of Cramer's rule to find the coordinates of an optimum vertex will yield integral answers. Hoffman [52–54] has given a characterization of *all* combinatorial theorems of this type, but the relevance of Cramer's rule to such theorems does not end there. One reason is that the relevant totally unimodular matrix may take some trouble to find, as in some generalizations of Dilworth's theorem [55]. Also, it may happen that, even though A is not totally unimodular, the right-hand side b has the property that every vertex of $\{x: Ax \leq b, x \geq 0\}$ is integral. For example, in the case of the so-called *matching* polytope, one can show that every vertex has the property that one of the relevant determinants for a Cramer's rule argument has determinant ± 1 [56].

Sometimes one can show that, for every integral c for which a dual problem $\max\{by: A^T y \leq c, y \geq 0\}$ has a solution, it has an integral solution. This implies [57] that every vertex of $\{x: Ax \geq b, x \geq 0\}$ is integral, so combinatorial interpretations follow. This fact can be used to prove generalizations of the famous "max flow = min cut" theorem [58]), the "shortest path = max cut packing" theorem [59], and matroid intersection results. For some of these, one shows that an optimal solution to the dual can be artfully recast from a solution to a problem using a totally unimodular matrix. For some others, the concept of lattice polyhedron [58, 60] yields a unification of several earlier results; and here the integrality of the dual problems follows from the existence of an optimal y whose support corresponds to rows forming a totally unimodular matrix. For other results (e.g., [61]), dealing with Berge's perfect and balanced matrices, unimodularity seems to provide no clue. For a recent status report, see [62].

A celebrated combinatorial use of inequalities and total unimodularity is Baranyai's parallelism theorem which asserts that, if a set N has kh elements, then there are disjoint partitions of N, each partition consisting of subsets of exactly h elements, such that each subset with h elements occurs in exactly one partition. The special case $h = 2$ corresponds to the classic problem of arranging k matches on each of $2k - 1$ days so that a league of $2k$ teams can complete a round robin tournament. This special case is easily done; indeed, there is a formula for doing so; but for general h it was not known before Baranyai whether it could be done, and a formula is still unavailable. He uses a complicated inductive argument, in which the critical step of the induction depends on total unimodularity, to prove the existence of the desired family of partitions. Building on Baranyai's approach, and using ideas of the cutting stock problem mentioned above, together with linear programming duality and some delicate computations involving binomial coefficients, one can find all pairs (n, h) such that if $|N| = n$, $h < n$, then there are disjoint partitions of N containing subsets of *at most* h elements, such that each S with $|S| \leq h$ occurs in exactly one partition [63, 64].

11. The 318-city problem

A TSP of 318 cities is not the largest problem ever tackled; we have heard of attempts made on problems of more than 600 cities. The 318-city problem may, or may not, be the largest problem ever solved; but it is the largest problem *known* to have been solved. It is the largest of the ten symmetric TSPs solved by Crowder and Padberg [65] in 1979. Three of those problems had already been solved, and solution proven; for the remaining problems, the optimal solutions were established for the first time. (Some of these had been previously solved, but optimality of the solutions had not been proved.)

As Crowder and Padberg state, their distant point of departure was the RAND work [13] of 1954; but their elaborate procedure, a sequel to that described in [66], used almost every device that has been mentioned so far.

They employed a procedure with three main steps. The first phase used the Lin-Kerningham [8] heuristic to find a good starting tour, and then a simplex-based procedure was applied to an LP problem composed initially of the degree-two constraints. During the course of this procedure, suitable subtour-elimination, 2-matching, and comb constraints were derived and used to augment the current LP problem. These derived constraints had the property that they did not exclude feasible tours, but the new augmented problem was a tighter LP relaxation of the TSP problem. When the constraint generation procedure could no longer identify candidate constraints, the augmented LP problem was solved to optimality, yielding a true lower bound on the minimum tour length of the TSP problem.

Phase two involved fixing a subset of variables in the TSP problem to either 0 or 1. This procedure, first employed in this context in [13], utilized the upper bound of the tour length from the Lin-Kerningham heuristic, and the solution value and reduced costs from the LP problem solved in phase one. For the 318-city TSP problem, this allowed more than 49 000 of the 50 403 variables to be fixed (most at zero, of course), thus reducing the problem to about three percent of its original number of variables.

In the third phase, the reduced problem was treated as a 0–1 IP problem, utilizing MPSX-MIP/370 for its solution. The 0–1 solution to this problem was either an optimal solution to the reduced TSP problem—this solution, taken in conjunction with the variables fixed at 1 in phase two, yielded an optimal solution for the original TSP problem—or defined a collection of subtours for the reduced problem. In the latter case, appropriate constraints to exclude the current 0–1 solution were generated and adjoined to the problem, which was then re-solved. For the 318-city

problem, the 0–1 LP problem required two such augmentations before yielding the optimal tour. The total CPU time (on the Thomas J. Watson Research Center's IBM 370/168, using VM/370 and MVS/TSO) was just under six minutes.

Crowder and Padberg conclude [65]: "We cannot report any failure of the proposed methodology . . . . In our view this fact points to the suitability of facet-defining cutting-planes for the purpose of proving optimality in hard and difficult combinatorial optimization problems . . . . We are confident that problems involving 1000 cities and more are amenable to exact solution by today's technology."

Acknowledgment

We are particularly indebted to Harlan P. Crowder of the IBM Thomas J. Watson Research Center and Carlton E. Lemke of Rensselaer Polytechnic Institute, visiting the Center, for a great deal of help and counsel.

References

1. Karl Menger, *Ergebnisse eines Kolloquiums* **3,** 11–12 (1930).
2. M. M. Flood, "The Traveling-Salesman Problem," *Oper. Res.* **4,** 61–75 (1956).
3. R. E. Gomory, "The Traveling Salesman Problem," *Proceedings of IBM Scientific Computing Symposium on Combinatorial Problems, March 1964*, IBM Data Processing Division, White Plains, NY, March 1966, pp. 93–121.
4. M. R. Garey and D. S. Johnson, *Computers and Intractability*, W. H. Freeman & Co., San Francisco, 1979.
5. R. M. Karp, "Reducibility Among Combinatorial Problems," *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, Eds., Plenum Publishing Co., New York, 1972, pp. 85–103.
6. M. Held and R. M. Karp, "A Dynamic Programming Approach to Sequencing Problems," *SIAM J. Appl. Math.* **10,** 213–242 (1962).
7. M. Held and R. M. Karp, "The Travelling Salesman and Minimum Spanning Trees, Part I," *Oper. Res.* **18,** 1138–1162 (1970); "Part II," *Math. Program.* **1,** 6–26 (1971).
8. S. Lin and B. W. Kernighan, "An Effective Heuristic Algorithm for the Travelling-Salesman Problem," *Oper. Res.* **21,** 498–516 (1973).
9. R. M. Karp, "Probabilistic Analysis of Partitioning Algorithms for the Traveling-Salesman Problem in the Plane," *Math. Oper. Res.* **2,** 209–224 (1977).
10. George B. Dantzig, *Linear Programming and Extensions*, Princeton University Press, Princeton, NJ, 1963.
11. W. Orchard-Hays, "The Challenge of Analytic Use of Computers for Global Problems," *Computers and Mathematical Programming*, W. W. White, Ed., U.S. Department of Commerce, U.S. Government Printing Office, Washington, DC, 1978.
12. *IBM Mathematical Programming System Extended/370 (MPSX/370) Program Reference Manual*, Order No. SH19-1095, 1978; available through IBM branch offices.
13. G. B. Dantzig, D. R. Fulkerson, and S. M. Johnson, "Solution of a Large-Scale Travelling-Salesman Problem," *Oper. Res.* **2,** 393–410 (1954).
14. R. E. Gomory, "Outline of an Algorithm for Integer Solutions to Linear Programs," *Bull. Amer. Math. Soc.* **64,** 275–278 (1958).
15. R. E. Gomory, "Solving Linear Programming Problems in Integers," *Proc. Symp. Appl. Math.* **10,** 211–215 (1960).
16. R. E. Gomory, "An Algorithm for Integer Solutions to Linear Programs," *Recent Advances in Mathematical Programming*, R. L. Graves and P. Wolfe, Eds., McGraw-Hill Book Co., Inc., New York, 1963, pp. 269–302.
17. R. E. Gomory, "Large and Non-Convex Problems in Linear Programming," *Proc. Symp. Interact Math. Res. High-Speed Comp.* (American Mathematical Society) **XV,** 125–139 (1963).
18. P. Wolfe, "Convergence Theory in Nonlinear Programming," *Integer and Nonlinear Programming*, J. Abadie, Ed., North-Holland Publishing Co., Amsterdam, 1970, pp. 1–36.
19. G. B. Dantzig and P. Wolfe, "The Decomposition Algorithm for Linear Programming," *Econometrica* **29,** 767–778 (1961).
20. R. E. Gomory and T. C. Hu, "An Application of Generalized Linear Programming to Network Flows," *SIAM J. Appl. Math.* **10,** 260–283 (1962).
21. P. C. Gilmore and R. E. Gomory, "A Linear Programming Approach to the Cutting Stock Problem," *Oper. Res.* **9,** 849–859 (1961); "A Linear Programming Approach to the Cutting Stock Problem—Part II," *Oper. Res.* **11,** 863–888 (1963).
22. P. C. Gilmore and R. E. Gomory, "The Theory and Computation of Knapsack Functions," *Oper. Res.* **14,** 1045–1074 (1966).
23. R. E. Gomory, "On the Relation Between Integer and Non-Integer Solutions to Linear Programs," *Proc. Nat. Acad. Sci.* **53,** 260–265 (1965).
24. R. E. Gomory, "Faces of an Integer Polyhedron," *Proc. Nat. Acad. Sci.* **57,** 16–18 (1968).
25. R. E. Gomory, "Properties of a Class of Integer Polyhedra," *Integer and Non-linear Programming*, J. Abadie, Ed., North-Holland Publishing Co., Amsterdam, 1970, pp. 353–365.
26. R. E. Gomory, "Some Polyhedra Related to Combinatorial Problems," *J. Lin. Alg. Appl.* **2,** 451–558 (1969).
27. R. E. Gomory and E. L. Johnson, "Some Continuous Functions Related to Corner Polyhedra," *Math. Program.* **3,** Part I: 23–85, Part II: 359–389 (1972).
28. R. E. Gomory and E. L. Johnson, "The Group Problem and Subadditive Functions," *Mathematical Programming*, T. C. Hu and S. M. Robinson, Eds., Academic Press, Inc., New York, 1973, pp. 157–184.
29. E. L. Johnson, "On the Group Problem for Mixed Integer Programming," *Mathematical Programming Study 2: Approaches to Integer Programming*, M. L. Balinski, Ed., North-Holland Publishing Co., Amsterdam, 1974, pp. 137–179.
30. H. P. Crowder and E. L. Johnson, "Use of Cyclic Group Methods in Branch and Bound," *Mathematical Programming*, T. C. Hu and S. M. Robinson, Eds., Academic Press, Inc., New York, 1973, pp. 213–226.
31. E. L. Johnson, "Faces of Polyhedra for Mixed Integer Programming Problems," *Symp. Math.* **19,** 289–299 (1976).
32. C.-A. Burdet and E. L. Johnson, "A Subadditive Approach to Solve Linear Integer Programs," *Ann. Disc. Math.* **1,** 117–143 (1977).
33. E. L. Johnson, "Subadditive Lifting Methods for Partitioning and Knapsack Problems," *J. Algorithms* **1,** 75–96 (1980).
34. M. W. Padberg, "Packing, Covering, and Knapsack Problems," *Ann. Disc. Math.* **4,** 265–287 (1979).
35. M. Grötschel and M. W. Padberg, "On the Symmetric Travelling Salesman Problem," *Math. Program.* **16,** Part I: Inequalities, 265–280; Part II: Lifting Theorems and Facets, 281–302 (1979).
36. V. Chvatal, "Edmonds Polytopes and Weakly Hamiltonian Graphs," *Math Program* **5,** 29–40 (1973).
37. J. D. C. Little, K. G. Murty, D. W. Sweeney, and C. Karel, "An Algorithm for the Traveling Salesman Problem," *Oper. Res.* **11,** 972–989 (1963).
38. K. Spielberg, "Algorithms for the Simple Plant Location Problem With Some Side Conditions," *Oper. Res.* **17,** 85–111 (1969).
39. K. Spielberg, "Plant Location With Generalized Search Origin With Some Side Conditions," *Manage. Sci.* **16,** 165–178 (1969).
40. C. E. Lemke, H. M. Salkin, and K. Spielberg, "Set Covering By Single Branch Enumeration with LP Subproblems," *Oper. Res.* **19,** 998–1022 (1971).
41. M. Guignard and K. Spielberg, "Reduction Methods for State Enumeration Integer Programming," *Ann. Disc. Math.* **1,** 273–285 (1977).

42. M. Guignard and K. Spielberg, "A Direct Dual Method for the Mixed Plant Location Problem With Some Side Conditions," *Math. Program.* **17,** 198–228 (1979).
43. *Mixed Integer Programming/370 (MIP/370) Program Reference Manual,* Order No. SH19-1099, 1979; available though IBM branch offices.
44. M. Benichou, J. M. Gauthier, P. Girodet, G. Hentges, G. Ribière, and O. Vincent, "Experiments in Mixed-Integer Linear Programming," *Math. Program.* **1,** 76–94 (1971).
45. R. T. Rockafellar, *Convex Analysis*, Princeton University Press, Princeton, NJ, 1970.
46. Yu. M. Ermolev and N. Z. Shor, "On the Minimization of Non-Differentiable Functions," (Russian) *Kibernetika* **3,** 101–102 (1967).
47. N. Z. Shor, "Cut-Off Method with Space Extension in Convex Programming Problems," (Russian) *Kibernetika* **13,** 94–95; translated as *Cybernetics* **13,** 94–96 (1977).
48. H. P. Crowder, M. Held, and P. Wolfe, "Validation of Subgradient Optimization," *Math. Program.* **6,** 62–88 (1974).
49. J. K. Cullum, W. E. Donath, and P. Wolfe, "An Algorithm for Minimizing Certain Non-Differentiable Convex Functions," [50] 35–55 (1975).
50. P. Wolfe, "A Method of Conjugate Subgradients for Minimizing Nondifferentiable Functions," [50] 145–173 (1975).
51. M. L. Balinski and P. Wolfe, Eds., *Mathematical Programming Study 3: Nondifferentiable Optimization,* North-Holland Publishing Co., Amsterdam, 1976.
52. A. J. Hoffman, "Total Unimodularity and Combinatorial Theorems," *Lin. Alg. Appl.* **13,** 103–108 (1976).
53. A. J. Hoffman, "Some Recent Applications of the Theory of Linear Inequalities to Extremal Combinatorial Analysis," *AMS Proc. Symp. Appl. Math.* **10** (*Combinatorial Analysis*), 113–127 (1960).
54. A. J. Hoffman, "The Role of Unimodularity in Applying Linear Inequalities to Proving Combinatorial Theorems," *Ann. Disc. Math.* **4,** 73–84 (1979).
55. A. J. Hoffman and H. S. Schwartz, "On Partitions of a Partially Ordered Set," *J. Comb. Theory* **B23,** 3–13 (1977).
56. A. J. Hoffman and R. Oppenheim, "Local Unimodularity in the Matching Polytope," *Ann. Disc. Math.* **2,** 201–209 (1978).
57. A. J. Hoffman, "A Generalization of Max Flow-Min Cut," *Math. Program*, **6,** 352–359 (1974).
58. R. E. Gomory and T. C. Hu, "Multi-Terminal Network Flows," *Siam J. Appl. Math.* **9,** 551–570 (1961).
59. E. L. Johnson, "On Cut-Set Integer Polyhedra," *Cahiers Centre Etudes Recherche Operationelle* **17,** 235–251 (1975).
60. A. J. Hoffman, "Blockers and Anti-Blockers of Lattice Clutters," *Mathematical Programming Study 8: Polyhedral Combinatorics*, M. L. Balinski and A. J. Hoffman, Eds. North-Holland Publishing Co., Amsterdam, 1978, pp. 197–207.
61. D. R. Fulkerson, A. J. Hoffman, and R. Oppenheim, "On Balanced Matrices," *Mathematical Programming Study 1: Pivoting and Extensions*, M. L. Balinski, Ed., North-Holland Publishing Co., Amsterdam, 1974, pp. 120–133.
62. A. J. Hoffman, "Linear Programming and Combinatorics," *Proc. Symp. Pure Math.* **34,** 245–253 (1979).
63. C. Berge and E. L. Johnson, "Coloring the Edges of a Hypergraph and Linear Programming Techniques," *Research Report CORR 76/4*, Dept. of Combinatorics and Optimization, University of Waterloo, Ontario, 1976.
64. E. L. Johnson, D. Newman, and K. Winston, "An Inequality on Binomial Coefficients," *Ann. Disc. Math.* **2,** 155–159 (1978).
65. H. Crowder and M. W. Padberg, "Solving Large-Scale Symmetric Traveling Salesman Problems to Optimality," *Manage. Sci.* **26,** 495–509 (1980).
66. M. W. Padberg and S. Hong, "On the Symmetric Travelling Salesman Problem: A Computational Study," *Mathematical Programming Study 12: Combinatorial Optimization*, M. W. Padberg, Ed., North-Holland Publishing Co., Amsterdam, 1980, pp. 78–107.

Received October 6, 1983

Michael Held *IBM Academic Information Systems, 1 North Broadway, White Plains, New York 10601.* Mr. Held joined IBM in 1958; he held positions in several groups involved with IBM's early efforts in scientific computing, including the New York Scientific Center and the Systems Research Institute, where he taught courses and conducted research in mathematical programming and computational complexity. He was an adjunct professor of operations research and industrial engineering at Columbia University and held a variety of editorial positions, including coeditorship of *Mathematical Programming* and *Management Science.* Mr. Held served on the Council of the Mathematical Programming Society, was Chairman of its Publications Committee, and is currently Chairman of its Executive Committee.

Alan J. Hoffman *IBM Research Division, P.O. Box 218, Yorktown Heights, New York 10598.* Dr. Hoffman is manager of senior consultants in the Mathematical Sciences Department of the Thomas J. Watson Research Center. His research interests include combinatorics, linear algebra, and linear programming. Prior to joining IBM in 1961, he was a member of the Institute for Advanced Study, a mathematician at the National Bureau of Standards, a scientific liaison officer with the Office of Naval Research (London), and a consultant with the General Electric Company. He received his Ph.D. in mathematics from Columbia University, where he was also an undergraduate, in 1950. He is a Founding Editor of *Linear Algebra and its Applications*, and is on the editorial boards of *Discrete Mathematics, Discrete Applied Mathematics, Naval Research Logistics Quarterly* and *Combinatorica.* He is an IBM Fellow, a member of the National Academy of Sciences, and a Fellow of the New York Academy of Science

Ellis Lane Johnson *IBM Research Division, P.O. Box 218, Yorktown Heights, New York 10598.* Dr. Johnson is manager of the mathematical programming and modeling group in the Mathematical Sciences Department of the Thomas J. Watson Research Center, which he joined in 1968. His principal work has been on the theory, computational methods, and applications of integer programming. He received a B.S. in mathematics from Georgia Institute of Technology in 1960 and the Ph.D. in operations research at the University of California at Berkeley in 1965. He was an assistant professor at Yale University from 1964 to 1967, and has had visiting appointments at the University of Waterloo, Ontario, the University of Bonn, Federal Republic of Germany, and the University of Pisa, Italy. He is an editor for *Discrete Applied Mathematics* and the *SIAM Journal of Algebraic and Discrete Mathematics.*

Philip Wolfe *IBM Research Division, P.O. Box 218, Yorktown Heights, New York 10598.* Dr. Wolfe is a senior consultant in the Mathematical Sciences Department of the Thomas J. Watson Research Center. His research interests are centered in the subject of mathematical programming. He received the A.B. in mathematics and physics in 1948, and the Ph.D. in mathematics in 1954, from the University of California at Berkeley. Subsequently he taught at Princeton University, then was a mathematician at the RAND Corporation, Santa Monica, California, from 1957 to 1966, when he joined IBM. He was a founder of the Mathematical Programming Society, which he served as chairman from 1978 to 1980, the Friends of Optimization, and the journal *Mathematical Programming.* He is a Fellow of the American Association for the Advancement of Science and of the Econometric Society.