HISTORY OF MATHEMATICAL PROGRAMMING

A Collection of Personal Reminiscences

EDITED BY

J.K. Lenstra
A.H.G. Rinnooy Kan
A. Schrijver

CWI
NORTH-HOLLAND

Early Integer Programming

Ralph E. Gomory

During the academic year 1953-54, I was a third year graduate student in mathematics at Princeton, doing research on nonlinear differential equations. I wrote several papers, the first of which became my PhD thesis. I was fortunate in having as my thesis advisor the remarkable and inspiring Professor Solomon Lefschetz. Armed with my PhD degree, I entered the US Navy in the fall of 1954 and spent four months at Officer Candidate School in Newport, Rhode Island. After that the Navy assigned me to the Physics Branch of the Office of Naval Research in Washington, and I arrived there early in 1955.

Down the hall from the Physics Branch was the Operations Research Group. By 1956 I knew some of the people there and had learned something about what operations research was. Also by that time I had learned my duties at the Physics Branch well enough so that I did them in less than full time, and Frank Isaakson, my helpful Branch Head, permitted me to spend my spare time with the Operations Research Group.

I had always wanted to try applied mathematical work, and the time I spent with the Operations Research Group looking at various Navy weapons systems strengthened that interest. I decided tentatively to make operations research my future work, and by way of preparation took, in 1957, an evening course in operations research given by Alan Goldman. This was my first encounter with linear programming.

Later in 1957, as the end of my three year tour of duty in the Navy was approaching, Princeton invited me to return as Higgins Lecturer in Mathematics. Because of my interest in applied work I had planned to look for an industrial position, but I decided instead to accept this attractive offer and spend a year or two at Princeton before going on.

When I returned to Princeton late in the fall of 1957, I got to know Professor A. W. Tucker, then the department head, who was the organizer and prime mover of a group interested in game theory and related topics. This group included Harold Kuhn and Martin (E. M. L.) Beale.

The Navy had kept me on as a consultant, so I continued to work on Navy problems through monthly trips to Washington. On one of these trips a group presented a linear programming model of a Navy Task Force. One of the presenters remarked that it would be nice to have whole number answers as 1.3 aircraft carriers, for example, meant nothing.

I thought about his remark and determined to try inventing a method that would produce integer results. I thought it was clearly important, as after all, indivisibilities are everywhere, but I also thought it should be possible. My view of linear programming was that it was the study of systems of linear inequalities and that it was closely analogous to studying systems of linear equations. Systems of linear equations could be solved in integers (diophantine equations), so why not systems of linear inequalities?

Returning to the office I shared with Bob Gunning (now Dean of the Faculty at Princeton), I set to work and spent about a week of continuous thought trying to combine methods for linear diophantine equations with linear programming. This produced nothing but a large number of partly worked out numerical examples and a huge amount of waste paper.

Late in the afternoon of the eighth day of this I had run out of ideas. Yet I still believed that, if I had to, in one way or another, I would always be able to get at an integer answer to any particular numerical example. At that point I said to myself, suppose you really had to solve some particular problem and get the answer by any means, what would be the first thing that you would do? The immediate answer was that as a first step I would solve the linear programming (maximization) problem and, if the answer turned out to be $7\frac{1}{4}$, then I would at least know that the integer maximum could not be more than 7. No sooner had I made this obvious remark to myself than I felt a sudden tingling in two of my left toes, and with great excitement realized that I had just done something different, and something that was not a part of classical diophantine analysis. How exactly had I managed to conclude, almost without thought, that, if the LP answer was $7\frac{1}{4}$, the integer answer was at most 7?

As I was working with equations having integer coefficients and only integer variables, it did not take me long to conclude that the reasoning involved two steps. First that the objective function was maximal on the linear programming problem and therefore as large or larger than it could ever be on the integer problem. Second that the objective function was an integer linear form and therefore had to produce integer results for any integer values of the variables, including the unknown integer answer.

Therefore the objective function had to be an integer less than $7\frac{1}{4}$, so it was legitimate to add the additional linear constraint, objective function $\leqslant 7$. I thought of this as 'pushing in' the objective function.

It was also immediately clear to me that there could be many other integer forms maximal at that vertex that could also be 'pushed in' in the same way. Greatly excited I set to work and within a few days had discovered how to generate maximal integer forms easily from the rows of the transformed simplex matrix. It became clear rapidly that any entry in a given row of the tableau could be changed by an integer amount while remaining an integer form, that these changes could be used to create a form that was maximal, as that simply meant that all the row entries had to become negative (in the sign convention I was then using). It also was clear that, once an entry became negative, it strengthened the new inequality if the entry was as small as possible in absolute value, so all coefficients were best reduced to their negative fractional parts. This was the origin of the 'fractional cut'.

Within a very few days, I had worked out a complete method using the fractional cuts. I thought of this method as 'The Method of Integer Forms'. With it I was steadily solving by hand one small numerical example after another and getting the right answer. However, I had no proof of finiteness.

I also observed that the fractional rows I was creating seemed to have a lot of special properties, all of which were explained later in terms of the factor group.

Just at this time I ran into Martin Beale in the hall. He was looking for a speaker for the seminar we had on game theory and linear programming. I said I would be glad to give a talk on solving linear programs in integers. Martin said 'but that's impossible'. That was my first indication that others had thought about the problem.

During the exciting weeks that followed, I finally worked out a finiteness proof and then programmed the algorithm on the E101, a pin board computer that was busy during the day but that I could use late at night. The E101 had only about 100 characters of memory and the board held only 120 instructions at one time, so that I had to change boards after each simplex maximization cycle and put in a new board that generated the cut, and then put the old board back to remaximize. It was also hard work to get the simplex method down to 120 E101 instructions. But the results were better and more reliable than my hand calculations, and I was able to steadily and rapidly produce solutions to four- and five-variable problems.

During these weeks I learned that others had thought about the problem and that George Dantzig had worked on the traveling salesman problem and had applied special handmade cuts to that.

Professor Tucker, who was enormously helpful during this period, as during my entire stay at Princeton, gave me the time he had for himself on the

program of a mathematical society meeting. There early in 1958 we made the first public presentation of the cutting plane algorithm [1, 2]. This produced a great deal of reaction, many people wrote to me, and Rand Corporation invited me to come out to California for the summer.

In the summer of 1958 I flew west to Los Angeles, where Rand was located, carrying the first edition of the manual for Fortran, then a brand new language. I spent one month at Rand and succeeded in producing a working version of the algorithm, written in Fortran, for the IBM 704. During my stay at Rand, I renewed my acquaintance of graduate student days with Lloyd Shapley and Herb Scarf (whom I now count as my oldest friend) and met for the first time George Dantzig, Dick Bellman, and Phil Wolfe. Phil, already well known for his work on quadratic programming, generously took on the assignment of orienting me during my visit at Rand. He helped me in every conceivable way and I am pleased that even today, more than thirty years later, we are still friends.

The Fortran program seemed to be debugged about two days before I left Rand so I was able to do larger examples. Larger meant something like ten to fifteen variables. Most of these problems ran quickly, but one went on and on and on producing reams of printout but never reaching a final answer. I thought at the time that perhaps there were still bugs left in the program, but in fact it was the first hint of the computational problems that lay ahead.

I had been interested in economics since my undergraduate days and had studied economics at night while in the Navy. Integer programming and the questions that it raised about pricing brought me together with William Baumol, then an Economics Professor at Princeton. Bill and I had an extremely interesting time working on [3], which was to my knowledge the first paper attempting to link integer programming and pricing.

In the summer of 1959, I joined IBM Research and was able to compute in earnest. Compute in earnest meant that we had one day turnaround on a 704. But that was enough because I also had a wonderful programmer, Carol Shanesy. We started to experience the unpredictability of the computational results rather steadily.

I then turned my mind to something I had always wanted to have, an all-integer programming algorithm, an algorithm that resembled the simplex method but in which the coefficients always remained integers. In this sense the method I was aiming at resembled the classical diophantine approach.

I was able at the time to invent such a method [4]. To keep all elements integer the pivot element had to be a 1 at all times. I did succeed in generating additional rows (valid inequalities) for the dual simplex method in such a way that there was always a candidate element for pivot element that was 1, and in such a way that it was always chosen by the normal rules of pivot choice. This was what I had more or less imagined and hoped for in the

earlier period when I was doing hand calculations, so I was tremendously delighted and excited. In addition I found that when the problem of finding the greatest common divisor of two numbers was formulated as an integer programming problem, the new method did it using the same steps as the classical methods.

However, when Carol Shanesy ran the early versions of the program, it was more or less uniformly inferior to the older method, and some of the numbers, while remaining integers, became very large. This poor computational performance was a great disappointment to me.

Over the next several years I was fortunate in having two outstanding collaborators. I worked with T. C. Hu on multi-terminal network flows [5] and some related problems, and with Paul Gilmore on the traveling salesman problem [6] and on many aspects of the cutting stock problem [7, 8, 10]. This last was real operations research and eventually won us the Lanchester Prize. Paul and I visited paper mills and glass plants, learned to understand what the cutting process was like there, and learned to run actual paper mill data. Sometimes, when our results were enough better than what was being done in the mills, the mill people would buy computers that took over the cutting process and saved paper.

Our main tool was linear (not integer) programming. We needed to deal with an enormous matrix whose columns were all possible ways to cut up rolls of paper. Generalizing the Dantzig-Wolfe decomposition, we developed [9] a technique that used the knapsack problem to generate implicitly all possible ways to cut paper and selected the best one using the shadow prices. This best cutting pattern was than added to the matrix and we then proceeded to the next step of the simplex method.

Since we did a rather large knapsack problem at each step of the simplex method, we were interested in developing and testing fast knapsack algorithms. Due to the work of Carol Shanesy, we were running and looking at a large amount of paper mill data. Looking at all this data, we started to see something peculiar about the solutions. This led rapidly to our discovery of the periodic nature of the solutions to large knapsack problems, something that was thrilling and quite unexpected to us at the time [11].

Given my past experience with integer programming, it was clear to me that there should be an n-dimensional integer programming analogue to what is after all the one-dimensional case, so I went to work on that. After some effort, this resulted in the theory of asymptotic integer programming and the discovery of the corner polyhedra, which I consider to be my best work in the field of integer programming [12, 13, 14].

Let me explain briefly what the corner polyhedra are. If you take any vertex of any linear programming polyhedron and drop all constraints except the N constraints forming that vertex, the remaining N constraints form a

cone. The convex hull of the integer points in that cone is the corner polyhedron for that vertex. We can also associate with that vertex a finite Abelian group, which is essentially the integers in N-space modulo the normal vectors of the N binding constraints.

All possible corner polyhedra for all possible linear programming problems turn out, surprisingly enough, to be faces of a single sequence of polyhedra called the master polyhedra. Each master polyhedron is associated with a finite Abelian group, a single master polyhedron for each group. As the groups get larger, the polyhedra get more complex rapidly. To a considerable extent, the polyhedra and their associated integer programming problems can be arranged in a natural order of increasing complexity. I tend to regard these polyhedra as the unavoidable atoms of integer programming.

The theoretical aspects of corner polyhedra have been worked on, but their use in computation has been pursued less. For example, the use of polyhedra to generate a natural sequence of test problems, or the use of the simpler polyhedra, or of the continuous versions mentioned below, as a means of approximation in large problems, remain to my knowledge possibilities that are largely unexplored.

That these areas have not been explored is one of my regrets about corner polyhedra. The other is that I did not have the wit at the time to name them the G (as in Gomory)-polyhedra.

About this time Ellis Johnson joined IBM Research and quite on his own became interested in this area of work. I had the pleasure of working with Ellis on a series of integer programming papers related to the corner polyhedra [15, 16, 17]. Unfortunately, our very fruitful collaboration came to an end when I became Director of Research for IBM in 1970, an event that ended my connection with integer programming at least till the present.

References

1. R. E. Gomory (1958). Outline of an algorithm for integer solutions to linear programs. *Bull. Amer. Math. Soc. 64*, 275-278.

2. R. E. Gomory (1963). An algorithm for integer solutions to linear programs. R. L. Graves, P. Wolfe (eds.). *Recent Advances in Mathematical Programming*, McGraw-Hill, New-York, 269-302.

3. R. E. Gomory, W. J. Baumol (1960). Integer programming and pricing. *Econometrica 28*, 521-550.

4. R. E. Gomory (1963). An all-integer integer programming algorithm. J. F. Muth, G. L. Thompson (eds.). *Industrial Scheduling*, Prentice-Hall, Englewood Cliffs, 193-206.

5. R. E. Gomory, T. C. Hu (1961). Multi-terminal network flows. *J. SIAM 9*, 551-570.

6. P. C. Gilmore, R. E. Gomory (1964). Sequencing a one state-variable

machine: a solvable case of the traveling salesman problem. *Oper. Res. 12*, 655-679.

7. P. C. Gilmore, R. E. Gomory (1961). A linear programming approach to the cutting-stock problem. *Oper. Res. 9*, 849-859.

8. P. C. Gilmore, R. E. Gomory (1963). A linear programming approach to the cutting-stock problem — Part II. *Oper. Res. 11*, 863-888.

9. R. E. Gomory (1963). Large and non-convex problems in linear programming. *Proc. ACM Symp. Interactions between Mathematical Research and High-Speed Computing XV*, 125-139.

10. P. C. Gilmore, R. E. Gomory (1965). Multistage cutting stock problems of two and more dimensions. *Oper. Res. 13*, 94-120.

11. P. C. Gilmore, R. E. Gomory (1966). The theory and computation of knapsack functions. *Oper. Res. 14*, 1045-1074.

12. R. E. Gomory (1965). On the relation between integer and noninteger solutions to linear programs. *Proc. Nat. Acad. Sci. 53*, 260-265.

13. R. E. Gomory (1967). Faces of an integer polyhedron. *Proc. Nat. Acad. Sci. 57*, 16-18.

14. R. E. Gomory (1969). Some polyhedra related to combinatorial problems. *Linear Algebra Appl. 2*, 451-558.

15. R. E. Gomory, E. L. Johnson (1972). Some continuous functions related to corner polyhedra. *Math. Programming 3*, 23-85.

16. R. E. Gomory, E. L. Johnson (1972). Some continuous functions related to corner polyhedra, II. *Math. Programming 3*, 359-389.

17. R. E. Gomory, E. L. Johnson (1973). The group problems and subadditive functions. T. C. Hu, S. M. Robinson (eds.). *Mathematical Programming*, Academic Press, New York, 157-184.